НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих*

Виконав: студент IV курсу, групи *ІП-63 Карпа Маркіян Володимирович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ас. Нечай Д.О.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім’я, по батькові (підпис)

Рецензент:

ас. каф. ТК Шемсединов Т.Г.
посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
11.06.2020 00:53:32 EEST

Дата звіту:
11.06.2020 22:59:07 EEST

ID перевірки:
1003947608

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Karpa_ip63

ID **файлу:** 1003962800 **Кількість сторінок:** 38 **Кількість слів:** 6406 **Кількість символів:** 48864 **Розмір файлу:** 75.33 KB

6.13% Схожість

Найбільша схожість: 1.69% **з джерело бібліотеки.** ID **файлу:** 1000020611

5.04% Схожість з Інтернет джерелами	60	Page 40
5.65% Текстові збіги по Бібліотеці акаунту	156	Page 40

0.78% Цитат

Цитати	1	Page 41

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Не знайдено замінених символів

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Карпі Маркіяну Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проекту «*Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих»*
керівник проекту *Нечай Дмитро Олександрович, ас.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджена наказом по університету від «07» травня 2020 р. №*1081-с*
2. Термін подання студентом проекту «08» червня *2020 року*
3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення
3) Аналіз якості програмного забезпечення та опис випробувань
4) Розгортання програмного забезпечення, керівництво користувача

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

1) Схема структурна варіантів використання

2) Схема структурна діяльності

3) Схема структурна класів програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « *10* » березня *2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*21.02.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*29.02.2020*	
3.	*Постановка та формалізація задачі*	*14.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*19.03.2020*	
5.	*Моделювання програмного забезпечення*	*02.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*10.04.2020*	
7.	*Розробка архітектури програмного забезпечення*	*15.04.2020*	
8.	*Розробка програмного забезпечення*	*21.04.2020*	
9.	*Налагодження програми*	*29.04.2020*	
10.	*Виконання графічних документів*	*12.05.2020*	
11.	*Оформлення пояснювальної записки*	*20.05.2020*	
12.	*Подання ДП на попередній захист*	*28.05.2020*	
13.	*Подання ДП рецензенту*	*01.06.2020*	
14.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Маркіян КАРПА
(підпис)

Керівник _____ Дмитро НЕЧАЙ
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6314.045440.02.81	Пояснювальна записка	60	
3	А4	КПІ.ІП-6314.045440.03.91	Технічне завдання	11	
4	А4	КПІ.ІП-6314.045440.04.13	Опис програми	48	
5	А4	КПІ.ІП-6314.045440.05.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-6314.045440.06.34	Керівництво користувача	18	
7	А3	КПІ.ІП-6314.045440.07.99	Схема структурна варіантів використання	1	
8	А3	КПІ.ІП-6314.045440.08.99	Схема структурна діяльності	1	
9	А3	КПІ.ІП-6314.045440.09.99	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6314.045440.00.90		
Розробн.		Карпа М.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Нечай Д.О.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проект складається з чотирьох розділів, містить 40 таблиць, 4 додатки, 10 джерел – загалом 150 сторінок.

Об’єкт дослідження: мобільний застосунок, що призначений для автоматизації створення HTTP API запитів.

Мета дипломного проекту: підвищення ефективності, спрощення та автоматизація виконання HTTP API запитів з мобільного пристрою з операційною системою на базі «Android», мінімізація помилок при формуванні відповідного запиту.

У першому розділі проведено аналіз предметної області, розглянуто відомі технічні рішення, їхні переваги та недоліки, а також сформульовано функціональні та нефункціональні вимоги до програмного забезпечення.

У другому розділі змодельовано мобільний застосунок, використовуючи структурні схеми бізнес-процесів, розглянуто архітектуру програмного забезпечення, описано програмні засоби, що використовувались при створенні додатку, проаналізовано безпеку даних.

У третьому розділі було проведено аналіз якості програмного забезпечення, та детально описано процес тестування.

У четвертому розділі описано процес розгортання мобільного додатку, та хід роботи з даним програмним забезпеченням.

У додатках наведено технічне завдання, керівництво користувача, схема структурна варіантів використання, програма та методика тестування.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, АВТОМАТИЗАЦІЯ, HTTP API ЗАПИТ, ВІДПОВІДЬ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 40 tables, 4 annexes, 10 sources – total 150 pages.

The object of study: mobile application, which can be used to automate the process of creating HTTP API requests.

The aim of the diploma project: to increase efficiency, simplify and automate the process of creating HTTP API requests from mobile device with operating system based on “Android”, to minimize the number of errors while creating such requests.

In the first section, the subject area was analyzed, existing technical solutions with their advantages and disadvantages were investigated, and functional and nonfunctional requirements for creating software application were formulated.

In the second section, the mobile application was modeled using structural BPMN diagrams, considered architecture of the application, described software used to create the application, analyzed data security.

In the third section, a software quality analysis was performed and the testing process was described in detail.

In the fourth section, describes the process of deploying a mobile application and presents how to work with this software.

Annexes contain the technical task, the users guide, the structural diagram of use-cases and testing program and methods.

KEYWORDS: MOBILE APPLICATION, AUTOMATION, HTTP API REQUEST, RESPONSE

Пояснювальна записка
до дипломного проєкту

на тему: Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP (від англ. Hyper Text Transfer Protocol) – це протокол прикладного рівня для передачі даних.

API (від англ. Application Programming Interface) – це набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

APK (від англ. Android Package) – формат архівних файлів-додатків для операційної системи «Android».

REST (від англ. Representational State Transfer) – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

SDK (від англ. Software Development Kit) – набір із засобів розробки, утиліт і документації, який дозволяє розробникам створювати прикладні програми для певної платформи.

Токен – випадково згенерований набір символів, що використовується для автентифікації користувача.

BPMN (від англ. Business Process Model and Notation) – це система умовних позначень для моделювання бізнес-процесів.

UML (від англ. Unified Modeling Language) – це мова моделювання, що призначена для створення абстрактної моделі системи.

URL (від англ. Uniform Resource Locator) – стандартизована адреса певного ресурсу.

ВСТУП

Якщо раніше Інтернет був розкішшю, і не кожен міг собі позволити його використовувати, то сьогодні він є усюди. Чи то вдома, чи в кафе, чи на зупинці автобуса – всюди можна отримати доступ до Всесвітньої Павутини. А оскільки практично кожен носить із собою смартфон, виникає необхідність в розробці мобільних застосунків, що могли б спілкуватись із віддаленим сервером. Разом із цим необхідно мати засоби для тестування методів відповідного сервера саме з мобільної платформи. Адже нерідко може виникнути ситуація, коли запит успішно проходить і опрацьовується, будучи надісланим з персонального комп'ютера, проте, надіславши з мобільного пристрою, отримаємо помилку.

На мобільному ринку існує чимала кількість додатків, що дозволяють виконувати HTTP запити різного вигляду. В більшості вони надають повний контроль над формуванням запиту. Проте в цьому є як і перевага так і дуже значний недолік. Адже надзвичайно легко помилитись при спробі сформувати необхідний запит. Також вони не надають ніякої інформації про структуру відповідного контролера та його методів. Через це при помилці користувач може не розуміти, де трапилась дана помилка – на стороні сервера, чи клієнта.

Метою створення даної роботи є підвищення ефективності та спрощення виконання HTTP запитів з мобільного пристрою, мінімізація помилок при формуванні відповідного запиту.

Завданням даної роботи є створення програмного продукту, який би допомагав розробникам, тестувальникам чи будь якій інші людині в тому, щоб викликати запити на якийсь віддалений веб-сервер з мобільного девайсу.

Результатом роботи є мобільний додаток, який можна використовувати в будь-якій сфері, де потрібно надсилати запити на веб-сервер для отримання необхідної інформації.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Інтернет – це всесвітня система сполучених комп’ютерних мереж. За останніми оцінками більше половини людей всього світу мають доступ до Всесвітньої Павутини. Тут можна робити практично все, що заманеться. Наприклад: дивитись фільми, читати книги і статті, слухати музику, виконувати покупки, спілкуватись з іншими людьми і так далі. Сьогодні все більше людей використовує сервіси, які можна отримати в Інтернеті, саме з мобільних пристроїв. Це пояснюється тим, що доступ до мережі є практично всюди, а телефон, у свою ж чергу, завжди під рукою.

Виникає необхідність в створенні мобільних додатків, які могли б взаємодіяти з мережею. Найчастіше така взаємодія відбувається з якимось віддаленим сервером, що містить певну інформацію та надає послуги, які клієнт не може сам виконати. Коли клієнт (мобільний додаток) хоче отримати певну інформацію від сервера, він повинен надіслати запит, а сервер його прийме, обробить та надішле відповідь. Виникає необхідність в додатках, які могли б виконувати різноманітні запити на різні сервери з метою тестування, як працездатності самого сервера так і роботи мобільних додатків. Адже не рідко виникають ситуації, коли запит надісланий з ПК, проходить і успішно обробляється, а з мобільного девайсу – ні.

Мобільні додатки сильно відрізняються від десктопних. Тут значно важче набирати текст, адже не має можливості використовувати фізичну клавіатуру, значно менший екран, через що при великій кількості графічних об’єктів дуже важко зрозуміти, що необхідно робити. Тому важливо створити додаток, який би був максимально простим в користуванні та надавав усю інформацію про віддалений сервіс, на який необхідно надсилати запити.

1.2 Змістовний опис і аналіз предметної області

Клієнт серверна архітектура – це архітектурний шаблон програмного забезпечення, який є головним у створенні розподілених мережних застосунків і передбачає взаємодію та обмін між ними. Як випливає з назви даний шаблон передбачає такі компоненти:

– набір серверів, що володіють інформацією, та надають певні послуги;

– набір клієнтів, що використовують сервіси, що надають сервери;

– мережа, що забезпечує взаємодію між серверами і клієнтами.

Що сервери, що клієнти є незалежними один від одного і між собою. Не існує жорсткої прив’язки між ними. Так клієнт може одночасно працювати з декількома серверами. І водночас сервер здатний співпрацювати з багатьма клієнтами паралельно. На рисунку 1.1 зображено приклад клієнт-серверної архітектури.

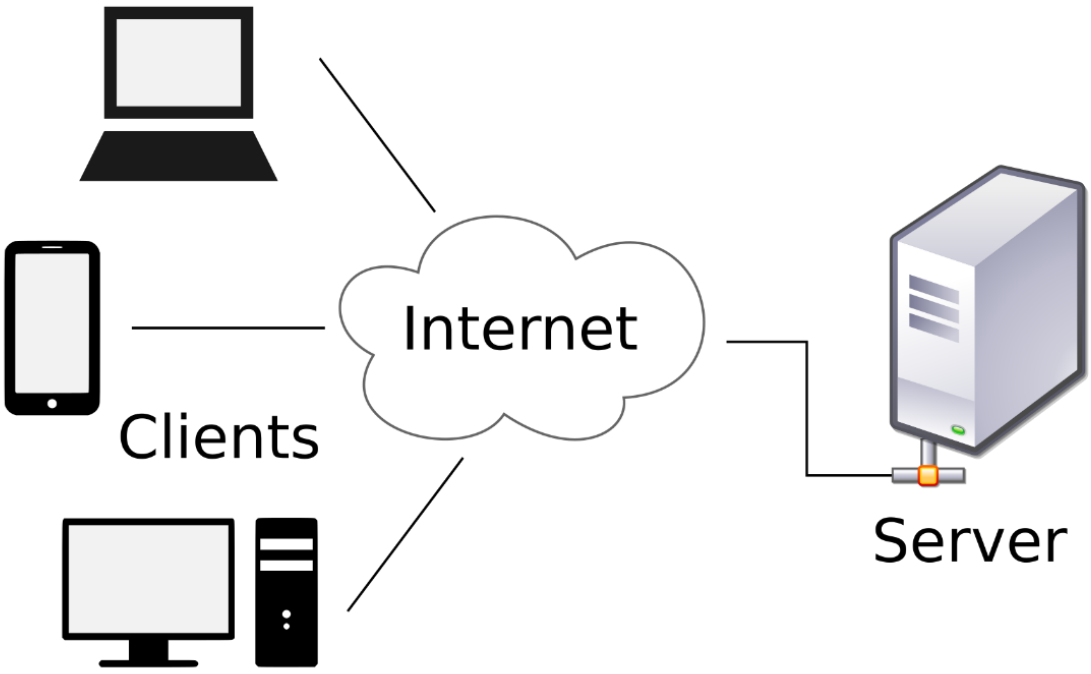


Рисунок 1.1 - Приклад клієнт-серверної архітектури

Клієнт та сервер спілкуються між собою за допомогою Запит-Відповідь

шаблону. Це означає, що коли в клієнта виникає необхідність в отриманні якоїсь інформації, він надсилає запит на сервер та очікує відповідь. Клієнт не знає, яким чином проводиться обробка цих даних, все про що він відає - це структура повідомлення для запиту . Сервер отримує запит від клієнта, обробляє його відповідним чином, та відсилає відповідь назад. Сервер нічогісінько не знає про клієнта, лише те, що прийшло йому в повідомленні запиту. Клієнт отримує відповідь, в якій повинна бути необхідна інформація. Для того, щоб клієнт мав змогу спілкуватись з сервером, вони повинні дотримуватись одних і тих же ж правил, які називають протоколами. Найпоширенішим протоколом, який використовують в клієнт-серверній архітектурі, та й в Інтернеті загалом є протокол HTTP.

HTTP – це протокол передачі даних, який використовується в комп'ютерних мережах. Кожен запит і відповідь складається з трьох частин:

- стартовий рядок;
- заголовок;
- тіло повідомлення.

Рядок запиту виглядає так: <Метод> <URI> HTTP/ <Версія>. Тут Метод – це метод HTTP запиту, URI – ідентифікатор ресурсу, версія – версія протоколу.

Заголовки – це набір пар ключ-значення, розділені двокрапками. Тут передається різноманітна службова інформація, наприклад: назва і версія браузера, адреса з якого прийшов клієнт і т.д.

Тіло повідомлення – це якраз самі дані, які необхідно передати.

Метод HTTP-запиту вказує серверу на те, яку дію необхідно виконати з певним ресурсом. Стандарт HTTP визначає 8 типів повідомлень. Найчастіше використовують 4 з них, а саме:

- GET – отримати представлення ресурсу;
- DELETE – видалити ресурс;
- POST – зберегти ресурс;

– PUT – оновити ресурс.

Методи GET, PUT та DELETE – ідемпотентні, що означає, що незалежно від того, скільки разів виконуєте операцію, яку вони просять, ви отримаєте той самий результат.

REST (Representational State Transfer) – це підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Обмеження:

– клієнт серверна архітектура;

– відсутність стану – це означає, що кожен запит містить усю необхідну інформацію для обробки, запити між собою не пов’язані, а отже не можлива бути така ситуація, коли якийсь запит містить інформацію з попереднього;

– кешування – означає, що дані, які передаються сервером повинні містити інформацію про те, чи можна їх кешувати. Це необхідно для того, щоб збільшити продуктивність, уникаючи не потрібних запитів;

– однорідний інтерфейс – усі компоненти повиннні підтримувати однорідний інтерфейс, адже це зменшує зв’язність між ними і сервісами. Ідентифікація ресурсів – URI повинен точно визначати ресурс, що очікується та якого формату має бути відповідь. Маніпуляція ресурсами через представлення – у випадку, коли клієнт містить представлення ресурсу, включаючи метадані, він має достатньо інформації, щоб модифікувати або ж видаляти цей ресурс;

– шари абстракції – кожен компонент потрапляє в якийсь шар і спілкується лише з компонентами в шарі під ним або в шарі над ним.

API (з англійської Application Programming Interface) – це набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Web API – це API для веб-серверів або веб-браузерів. Серверне web API – це програмний інтерфейс, який складається з одного чи багатьох публічних кінцевих точок. Кінцеві точки (англійською Endpoints) є надзвичайно

важливими в роботі Web API, адже вони визначають де знаходяться ресурси, які можуть бути досягнуті третьою стороною. Переважно доступ є через URI, до якого приходить запит, і від якого повинен надсилатись відповідь. Досить часто URI може мати наступний вигляд:

http://example.com/controller/method/{1}

Тут controller – це не що інше, як певний контейнер, який містить набір методів. Method – це функція, яка відповідає певному HTTP методу, та виконує якусь дію. Після методу, можна перераховувати набір значень, які будуть виступати в якості параметрів даного методу.

Swagger OpenAPI, яка початково була відомо як Swagger – це специфікація машиночитабельних файлів з інтерфейсами для опису, створення, візуалізації та використання REST веб-сервісів. Програмні забезпечення, які розробляються з допомогою файлів, що описують інтерфейси OpenAPI можуть автоматично генерувати документацію методів, параметрів та моделей. Специфікація OpenAPI не залежить від мови. З декларативною специфікацією ресурсу OpenAPI, клієнти можуть розуміти і використовувати сервіси без знання деталей реалізації сервера. Тобто, якщо сервер реалізує інтерфейс OpenAPI, клієнт має змогу отримати усю необхідну інформацію про контролери, методи та параметри цих методів і не тільки.

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих програмних продуктів

Найпопулярнішим програмним забезпеченням для створення HTTP запитів є Postman. Postman – це платформа для розробки і тестування API. Дане програмне забезпечення надає повний контроль над створенням запиту. Користувач має змогу ввести адресу необхідного веб-серверу, вибрати метод запиту, визначити дані, що будуть надходити в заголовку та в тілі повідомлення, і це лише малий список усього, що дозволяє даний застосунок. Словом, дає повний контроль над створенням запиту. В цьому є як перевага

так і основний недолік даного ПЗ. Річ у тім, що дозволяючи користувачу вводити будь-які дані в безліч полей, виникає велика імовірність помилитись, переважно через неуважність. У такому випадку регулярно виникають ситуації, коли не можливо зрозуміти, чи проблема на стороні сервера, чи в самому неправильно-сформованому запиті. Особливо це актуально, коли не можливо ніяким чином глянути на код серверної частини, маючи тільки клієнта. Також стосовно цього, користувач не має ніякого поняття, яким чином виглядає контролер, чи його методи, чи параметри всередині цих методів, тож імовірність помилки знову ж таки зростає. До всього цього потрібно додати, що Postman – це здебільшого програмний продукт для персональних комп'ютерів.

Для мобільних платформ можна знайти декілька аналогів Postman. Зокрема в магазині Android «Google Play», існують такі додатки:

– «Rest Client – Test REST API with your phone»;
– «Restler – REST API Client»;
– «HttpCanary – HTTP Sniffer/Capture/Analysis».

Усі вони мають функціональність схожу до Postman. Дозволяють користувачу повний контроль над створенням запиту, проте не дають жодної інформації про структуру даного API, тож імовірність помилки дуже висока. Враховуючи це, також слід добавити, що на мобільних пристроях значно складніше набирати дані ніж на ПК, використовуючи клавіатуру.

1.4 Аналіз вимог до програмного забезпечення

Для визначення вимог до програмного забезпечення необхідно вияснити ролі користувачів та їх можливості в системі. Система повинна містити наступні типи користувачів:

– користувач.

В програмному забезпеченні не буде передбачено створення адміністративних ролей тож кожен користувач буде мати однакову роль і зможе мати доступ до усієї функціональності.

1.4.1 Розроблення функціональних вимог

Варіант використання – це опис поведінки системи, як вона відповідає на зовнішні запити. Тобто описує «хто» і «що» може зробити з розглянутою системою.

Варіанти використання, які передбачені в даному застосунку наведено в таблицях 1.1 - 1.8.

Таблиця 1.1 - Варіант використання UC001

Назва	Отримання метаданих та побудова UI елементів на їх основі.
Опис	Користувач має можливість отримати інформацію про всі контролери, їхні методи та параметри даного API.
Учасники	Користувач.
Передумови	
Постумови	Користувачу відображається список усіх наявних контролерів відповідного API.
Основний сценарій	Система демонструє початкове вікно, яке містить поле для вводу адреси необхідного API, та кнопку підтвердження «Confirm». Користувач вводить адресу веб-сервера в полі для вводу та активує кнопку «Confirm». Після натискання, система надішле запит на отримання метаданих та, у разі успіху, обробить їх, і відкриє нову сторінку «Controllers», де буде відображено контролери.

Продовження таблиці 1.1

Розширення сценаріїв	Система виявляє, помилку, при виконанні запиту. Система сповіщає користувача про причину помилки.

Таблиця 1.2 - Варіант використання UC002

Назва	Перегляд створених методів.
Опис	Користувач має можливість переглянути список створених методів.
Учасники	Користувач.
Передумови	Користувач знаходиться на сторінці “Controller”.
Постумови	Користувачу відображається список методів для відповідного контролера.
Основний сценарій	Користувач натискає на потрібний контролер. Під даним контролером групою відображається список усіх його методів.
Розширення сценаріїв	

Таблиця 1.3 - Варіант використання UC003

Назва	Перегляд параметрів відповідного методу.
Опис	Користувач має можливість переглянути список параметрів відповідного методу.
Учасники	Користувач.
Передумови	Користувач знаходиться на сторінці “Controller”
Постумови	Користувачу відображається список параметрів для відповідного метода
Основний сценарій	Користувач натискає на потрібний контролер.

Продовження таблиці 1.3

	Користувач натискає на один із методів даного контролера. Під даним методом групою відображається список усіх його параметрів.
Розширення сценаріїв	

Таблиця 1.4 - Варіант використання UC004

Назва	Введення даних у залежності від типу параметра.
Опис	Користувач має можливість вводити дані відповідним чином у залежності від типу параметра.
Учасники	Користувач.
Передумови	Користувач знаходиться на сторінці “Controller”, відкривши певний контролер, а в ньому певний метод.
Постумови	Користувач вводить дані для заповнення параметрів метода.
Основний сценарій	Для текстових даних користувач вводить дані в текстове поле, для булевих значень – у відповідний Checkbox, для дати – у DatePicker.
Розширення сценаріїв	

Таблиця 1.5 - Варіант використання UC005

Назва	Валідація даних.
Опис	Користувач повинен мати можливість провести валідацію введених даних.

Продовження таблиці 1.5

Учасники	Користувач.
Передумови	Користувач відкрив певний метод, та заповнив параметри різними значеннями.
Постумови	У випадку, якщо дані є невалідними, відповідне повідомлення буде відображене користувачу.
Основний сценарій	Користувач ввів різні значення, не обов'язково правильні, в параметри метода. Після натискання кнопки «Execute» відбудеться валідація даних. Біля кожного невалідного значення буде відображено повідомлення. Наприклад (якщо в поле, яке передбачає введення числових значень ввести якесь текстове, під даним полем буде відображене повідомлення про помилку).
Розширення сценаріїв	

Таблиця 1.6 - Варіант використання UC006

Назва	Надсилання запиту на сервер.
Опис	Користувач повинен мати можливість надіслати запит на сервер.
Учасники	Користувач.
Передумови	Користувач відкрив певний метод, та заповнив параметри різними значеннями.
Постумови	Введені дані будуть відправлені на сервер в якості запиту.
Основний сценарій	Користувач, ввівши необхідні дані в параметри, натискає на кнопку «Execute».

Продовження таблиці 1.6

	У разі, якщо дані валідні, сформується запит і буде надісланий на веб сервер.
Розширення сценаріїв	

Таблиця 1.7 - Варіант використання UC007

Назва	Відображення відповіді від сервера.
Опис	Користувач повинен мати можливість отримати інформацію про відповідь від сервера.
Учасники	Користувач.
Передумови	Користувач натиснув на кнопку “Execute” та надіслав запит на сервер.
Постумови	Відповідь від сервера буде відображена у відповідному місці.
Основний сценарій	Відповідь з сервера приходить до користувача, і відображається в полі «Responses», описуючи статус повідомлення та його тіло.
Розширення сценаріїв	У випадку, якщо виникне помилка, відповідне повідомлення буде також відображено в полі «Responses».

Таблиця 1.8 – Варіант використання UC008

Назва	Введення токена авторизації
Опис	Користувач повинен мати можливість ввести токен авторизації.
Учасники	Користувач.
Передумови	Користувач відкрив сторінку «Controllers».
Постумови	Заданий токен авторизації.

Продовження таблиці 1.8

Основний сценарій	Користувач натискає на кнопку «Authorize». Після цього програма відображує модальне вікно, в якому можна задати токен. Користувач вводить токен авторизації у відповідне поле та натискає на кнопку підтвердження.
Розширення сценаріїв	

Функціональні вимоги – це вимоги до ПЗ, які описують внутрішню роботу системи. В даному застосунку було визначено такі функціональні вимоги, що зазначені в таблицях 1.9 – 1.20:

Таблиця 1.9 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Отримання метаданих
Опис	Надсилання запиту для отримання метаданих

Таблиця 1.10 - Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Формування об'єктів в пам'яті з отриманих метаданих
Опис	Після отримання метаданих, їх необхідно обробити та сформувати об'єкти відповідних класів

Таблиця 1.11 - Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Побудова UI елементів на основі об’єктів з метаданих

Продовження таблиці 1.11

Опис	Отримавши об'єкти класів, їх можна використати для побудови елементів графічного інтерфейсу

Таблиця 1.12 - Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Відкриття списку методів відповідного контроллера
Опис	При натисканні на певний контролер, список його методів повинен відобразитись

Таблиця 1.13 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Відкриття списку параметрів
Опис	При натисканні на певний метод, список його параметрів повинен відобразитись

Таблиця 1.14 - Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Побудова відповідних графічних елементів до типу та формату параметра
Опис	Різний тип, та різний формат параметра відповідає різному графічному об'єкту. Так, якщо елемент містить тип integer та формат enum, відобразиться випадаючий список, у випадку, якщо формату не буде вказано, відобразиться звичайне текстове поле для вводу.

Таблиця 1.15 - Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Валідація вхідних параметрів
Опис	При натисканні на кнопку «Execute» дані повинні валідуватись. У випадку, якщо параметр не правильний, відповідне повідомлення буде відображено біля даного значення

Таблиця 1.16 - Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Формування запиту
Опис	Після успішної валідації, усі дані будуть зібрані в один об'єкт для надсилання запиту

Таблиця 1.17 - Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Надсилання запиту
Опис	Після успішної валідації, та формування запиту, дані будуть надіслані на відповідний адрес

Таблиця 1.18 - Опис функціональної вимоги REQ010

Номер	REQ010
Назва	Отримання та відображення відповіді
Опис	Відображення відповіді від сервера в полі «Responses»

Таблиця 1.19 - Опис функціональної вимоги REQ011

Номер	REQ011
Назва	Демонстрація проводження певної роботи під час запиту
Опис	Для того, щоб користувач знав, що його запит обробляється, після натискання на кнопку «Execute» буде відображено індикатор виконання певного процесу. Після того, як прийде відповідь, даний індикатор заміниться самою відповіддю

Таблиця 1.20 – Опи функціональної вимоги REQ012

Номер	REQ012
Назва	Задання токену авторизації
Опис	Після того, як користувач введе токен авторизації у відповідне поле, цей токен буде збережено в пам'яті. Пізніше при створенні запиту він буде вказаний в якості значення до ключа «Authorization» в заголовку запиту

1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги описані в таблицях 1.21 – 1.24:

Таблиця 1.21 - Опис нефункціональної вимоги NFR001

Номер	NFR001
Назва	Операційна система Android 6.0 і вище
Опис	Додаток повинен підтримувати операційну систему Android 6.0 і вище

Таблиця 1.22 - Опис нефункціональної вимоги NFR002

Номер	NFR002
Назва	Зручність і простота
Опис	Застосунок повинен бути зручним і простим у користуванні

Таблиця 1.23 - Опис нефункціональної вимоги NFR003

Номер	NFR003
Назва	Швидкість запуску
Опис	Застосунок не повинен запускатись довше ніж 5 секунд

Таблиця 1.24 - Опис нефункціональної вимоги NFR004

Номер	NFR004
Назва	Швидкодія
Опис	Додаток не повинен зависати при одночасно відкритій великій кількості контролерів та методів

1.4.3 Постановка комплексу завдань модулю

Дане програмне забезпечення призначене для полегшення та прискорення роботи з HTTP API на мобільних платформах.

Метою створення даної роботи є підвищення ефективності та спрощення виконання HTTP запитів з мобільного пристрою, мінімізація помилок при формуванні відповідного запиту.

Для досягнення мети даної роботи система повинна вирішувати наступні задачі:

– отримання інформації про API певного сервісу;

– динамічна побудова списку контролерів даного сервісу;

– динамічна побудова списку методів відповідного контролера;

– динамічна побудова списку параметрів відповідного метода в залежності від типу та формату даного параметру;

– проведення валідації усіх введених параметрів;

– збирання введеної інформації та формування запиту;

– надсилання запиту на відповідну адресу;

– отримання та відображення відповіді від сервера.

Мобільний застосунок мусить працювати на пристроях з операційною системою на базі Android.

1.5 Висновки по розділу

Можливість виконання HTTP запитів з мобільних девайсів є надзвичайно важливою. Особливо в теперішньому світі, коли все більше і більше людей користується інтернетом і чи не кожен володіє смартфоном.

Існують додатки, які призначені для роботи з HTTP API запитами. Проте усі вони надають надто багато можливостей, при цьому, не надаючи інформації про сервіс, методи якого потрібно тестувати. Через це використовуючи дані аналоги дуже легко можна помилитись у формуванні повідомлення, і потім довго не розуміти, чи помилка виникла на сервері, чи проблема в самому запиті.

Для того, щоб максимально полегшити роботу зі створенням HTTP API запитів, було вирішено створити мобільний застосунок. Даний додаток буде відразу відображати усі контролери, методи цих контролерів та їхні параметри, буде проводити валідацію при спробі надіслати запит. Тож користувач відразу буде знати на який метод він відправляє повідомлення, чи такий метод узагалі існує, і які параметри, з яким типом та форматом необхідно надсилати.

В даному розділі було розглянуто можливі аналоги даного застосунку, розроблено функціональні та нефункціональні вимоги, а також було визначено комплекс завдань, які необхідно виконати для успішної роботи програми.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Щоб написати якісний застосунок необхідно спершу провести детальне моделювання та аналіз програмного забезпечення. Для проектування бізнес-процесів зручно використовувати методологію створення діаграм BPMN.

BPMN(з англійської Business Process Model and Notation) – це система умовних позначень для моделювання бізнес-процесів. Дана нотація дає можливість визначати складну семантику бізнес-процесів.

В системі існують три процеси, які проходить користувач, а саме:

– отримання метаданих та побудова на основі них графічних елементів;

– надсилання запиту на потрібну адресу та отримання відповіді;

– введення токена авторизації, який буде задаватись у заголовку кожного запиту.

Послідовний опис процесу отримання та обробки метаданих:

– на першій сторінці у відповідному полі для вводу користувач задає необхідну url-адресу;

– натискає на кнопку підтвердження;

– система надсилає запит на відповідну адресу та очікує відповідь;

– у випадку, якщо виникне помилка, додаток відобразить відповідне повідомлення;

– у випадку, якщо помилки не має, додаток обробить вхідні метадані;

– з оброблених даних сформуються графічні елементи, а саме список контролерів, їхніх методів, та параметрів усередині цих методів.

На рисунку 2.1 зображено схему, що описує даний бізнес-процес.

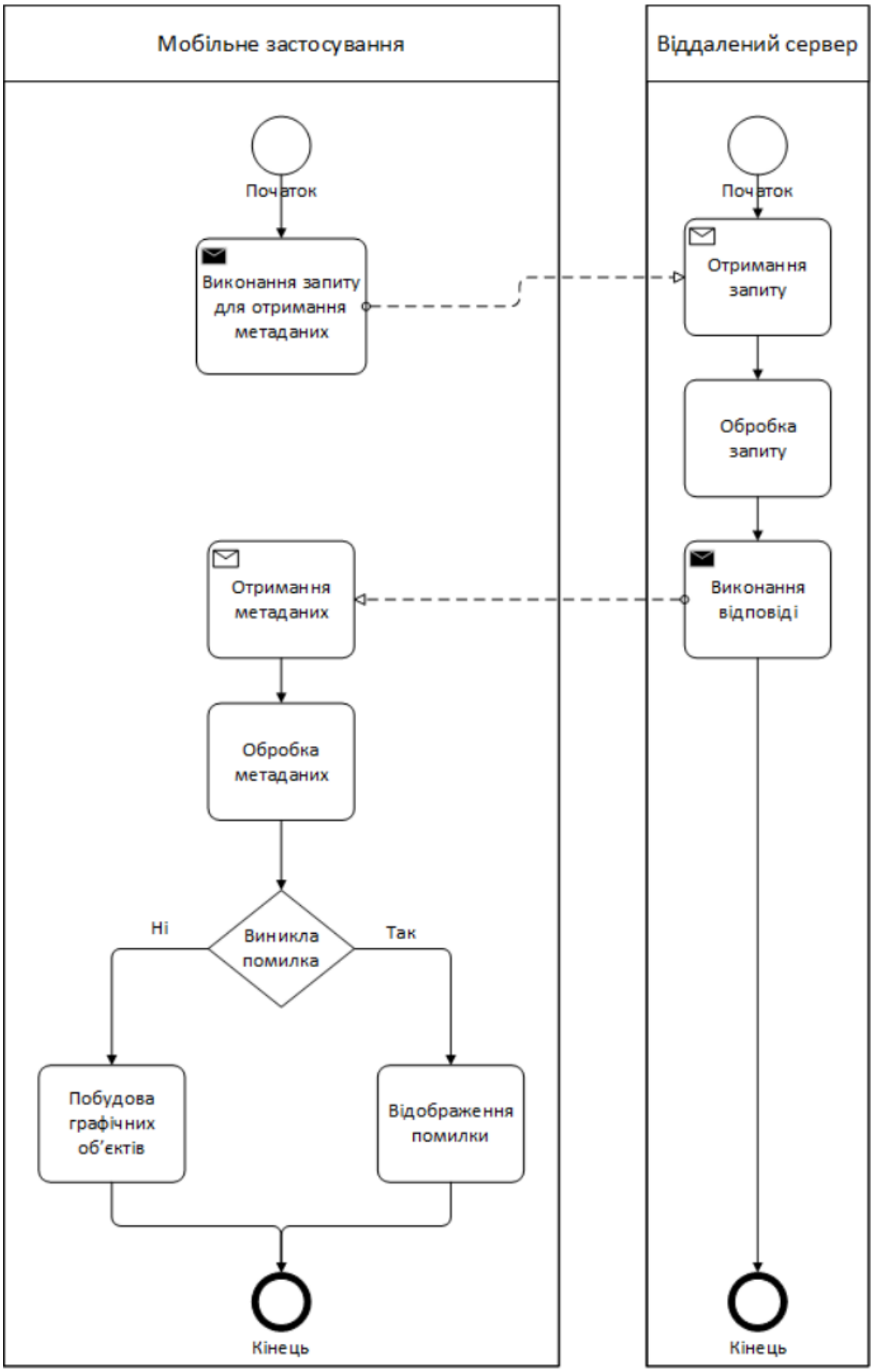


Рисунок 2.1 – Схема бізнес-процесу отримання та обробки метаданих

Послідовний опис процесу надсилання запиту та отримання відповіді:

– користувач заповнює параметри відповідного методу необхідними даними;

– користувач натискає на кнопку виконання запиту;

– застосунок проведе валідацію вхідних даних;

– у разі, якщо якісь параметри є не валідні, система виведе відповідне повідомлення, сповіщаючи користувача, про те, що необхідно внести певні зміни;

– у разі ж, коли введені дані є правильними, додаток надішле запит на відповідний веб-сервер та буде очікувати відповіді;

– якщо станеться помилка, або ж усе буде гаразд, в будь якому разі застосунок відобразить відповідне повідомлення в секції «Response».

На рисунку 2.2 зображено схему, що описує даний бізнес-процес.

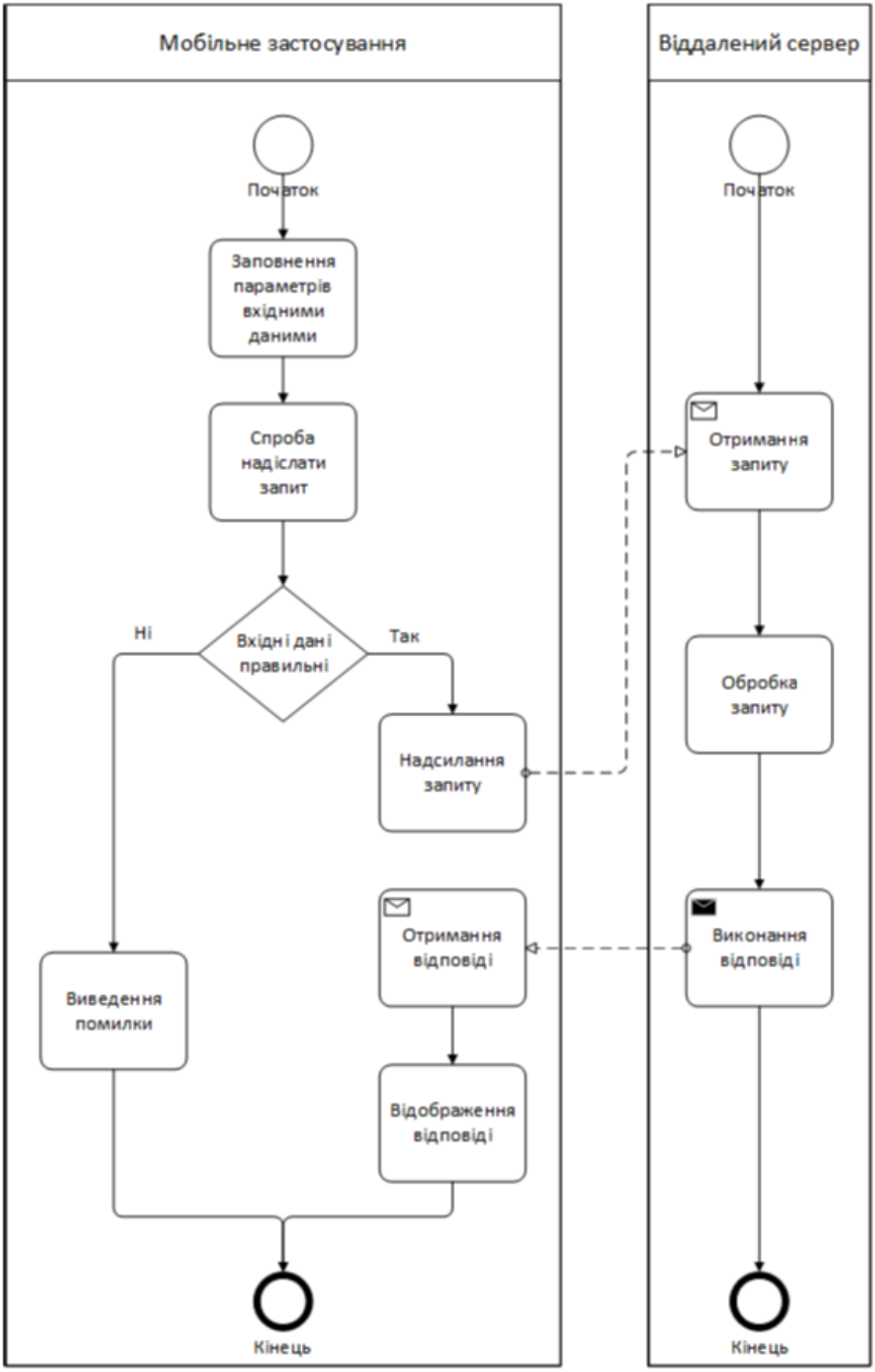


Рисунок 2.2 – Схема бізнес-процесу надсилання запиту

Якщо для виконання якого-небудь запиту необхідне підтвердження особистості персони, що виконує запит, на сторінці «Controllers» буде відображена кнопка «Authorize». Якщо натиснути на цю кнопку, користувачу буде відображено модальне вікно, в якому буде поле для вводу токена, детальна інформація, що описує даний токен та кнопки підтвердження і скасування операції. Користувач має змогу ввести відразу необхідний токен у відповідне поле. Після підтвердження даний токен буде збережений та використовуватиметься при кожному наступному запиті в заголовку повідомлення.

У випадку, якщо користувач не знає свого токена, він має змогу виконати запит на методи, що не потребують авторизації. Переважно це методи, що виконують автентифікацію користувача в системі. Якщо користувач успішно автентифікувався, йому буде повернено токен, який він може скопіювати і використовувати в кожному наступному запиті.

На рисунку 2.3 зображено схему, що описує даний бізнес-процес.

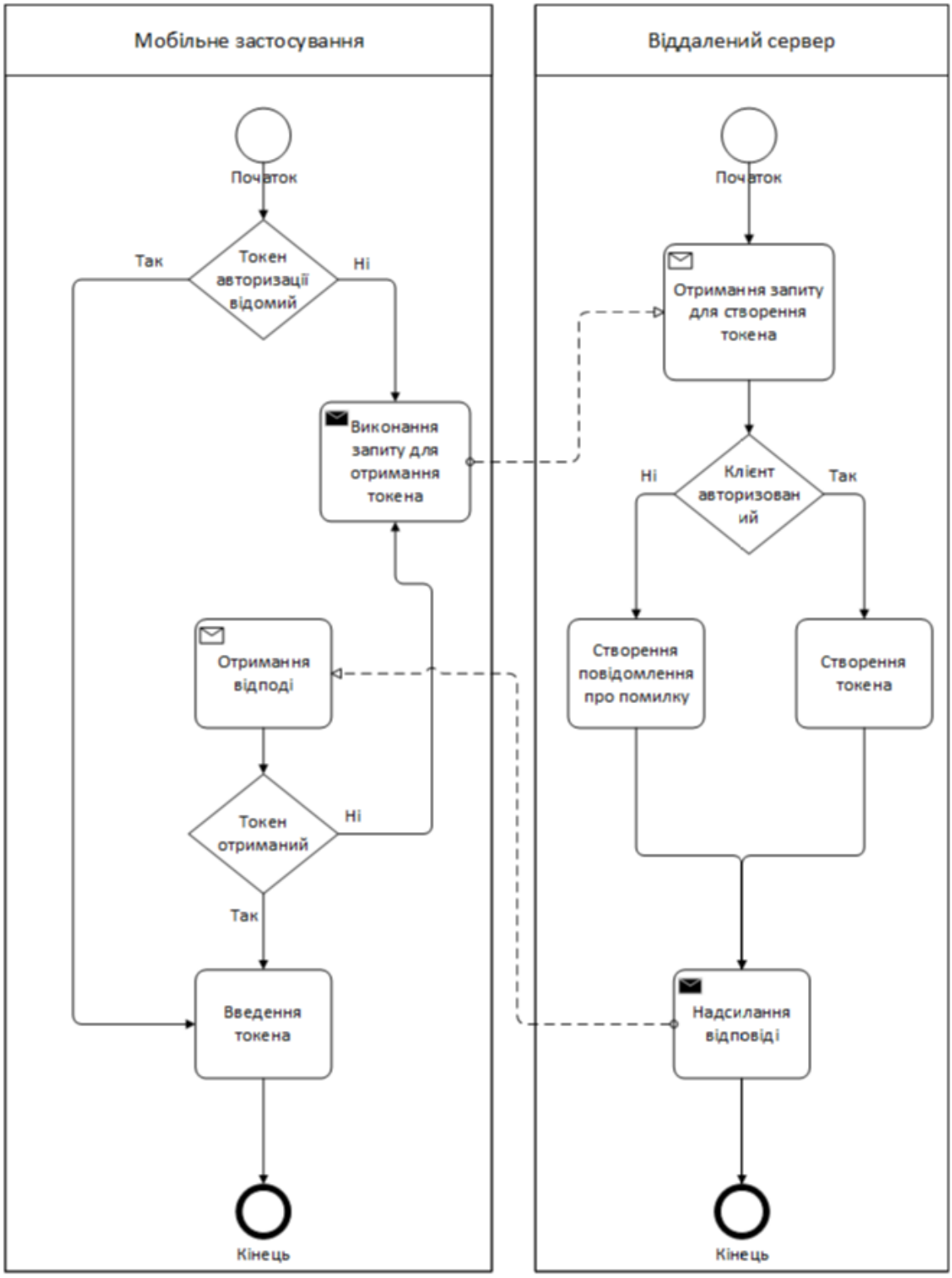


Рисунок 2.3 – Схема бізнес-процесу отримання токена авторизації

2.2 Архітектура програмного забезпечення

Для розробки мобільного застосунку необхідно перш за все вибрати програмний каркас. Для створення даного додатку було обрано Flutter від Google.

Flutter – це програмний каркас із відкритим кодом, для створення додатків для платформ Android та IOS, а також на вебі, розроблений компанією Google. В даного рушія є наступні переваги:

– перспективність і активний розвиток;

– детальна документація, як в текстовому так і в відео форматі;

– власний графічний рушій, що означає, що не має необхідності створювати інтерфейс окремо для Android і IOS;

– можливість виконувати «Hot Reload» - розробник може бачити свої зміни відразу, достатньо лише натиснути на кнопку оновлення.

Flutter складається з наступних елементів:

– Flutter рушій – програмний рушій для рендерингу, написаний в основному на C++ з використанням графічної бібліотеки Google Skia;

– базової бібліотеки (Foundation library) – бібліотека складається з класів та функцій, що написані на мові програмування Dart, які використовують для побудови Flutter програм, для взаємодії із Flutter рушієм;

– віджетів.

Архітектура Flutter відзрізняється від інших програмних каркасів, таких як: React, Cordova, тим, що він не використовує для побудови інтерфейсу мови HTML, CSS, Javascript, відповідно і вбудований рушій WebView. Використовується власний рушій для рендерингу. Основною мовою програмування даного каркасу є Dart.

Віджети – це базові будівельні блоки графічного інтерфейсу. Кожен віджет це незмінна декларація частини інтерфейсу користувача. На відміну від інших програмних каркасів, які розділяють Вид (View), Контролери Виду

(View Controllers), макет (Layout) та інші властивості, Flutter містить послідовний, уніфікований об’єкт – віджет.

Віджетом може бути:

– структурний елемент, на зразок кнопки, чи меню;

– елемент стилю, на зразок кольорової схеми чи шрифту;

– властивості макету, на зразок відступів усердині та ззовні об’єкта (padding, margin).

Віджети формують ієрархію. Кожен Віджет знаходиться всередині та уснаслідує властивості свого предка. Дуже часто віджети складаються з багатьох простіших віджетів, що виконують якусь одну функцію. На рисунку 2.3 зображено приклад ієрархії віджетів у Flutter.

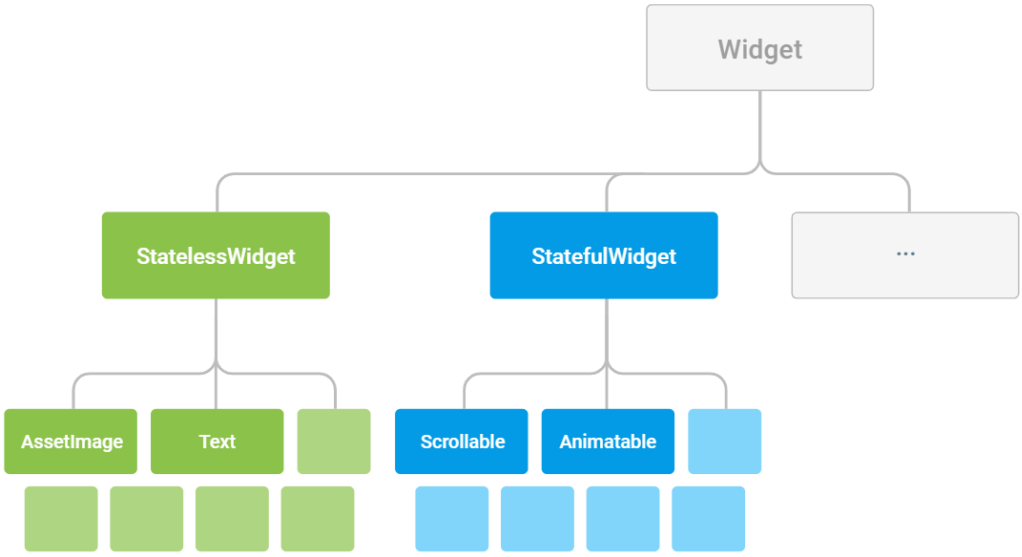


Рисунок 2.4 – Приклад ієрархії віджетів

При створенні нового власного віджету, його необхідно наслідувати від StatefulWidget або StatelessWidget. StatelessWidget - використовують тоді, коли необхідно відобразити елементи, що не будуть ніяким чином змінені в процесі виконання програми. Тобто він не має ніякої змоги перемальовуватись власноруч. Даний віджет є ефективнішим адже оперує лише з незмінними даними. На рисунку 2.4 зображена схема роботи StatelessWidget.

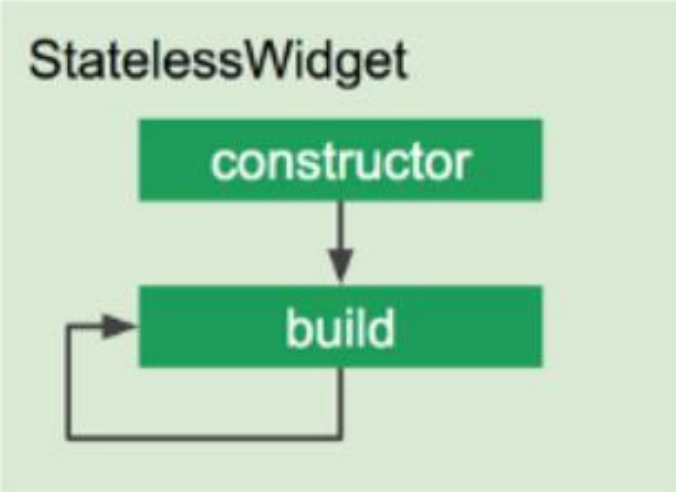


Рисунок 2.5 – Схема роботи StatelessWidget

StatefulWidget у свою ж чергу використовується завжди тоді, коли необхідно перебудовувати дерево віджетів. Наприклад при натисканні на кнопку, потрібно змінити якийсь текст. Для цього необхідно перебудувати дерево, а отже, потрібно використовувати StatefulWidget. Даний віджет не може існувати сам по собі, він завжди взаємодіє з класом State (стан) і містить метод createState, для створення даного стану. Коли необхідно виконати перебудову сцени, викликається метод setState. На рисунку 2.4 зображено схему роботи StatefulWidget.

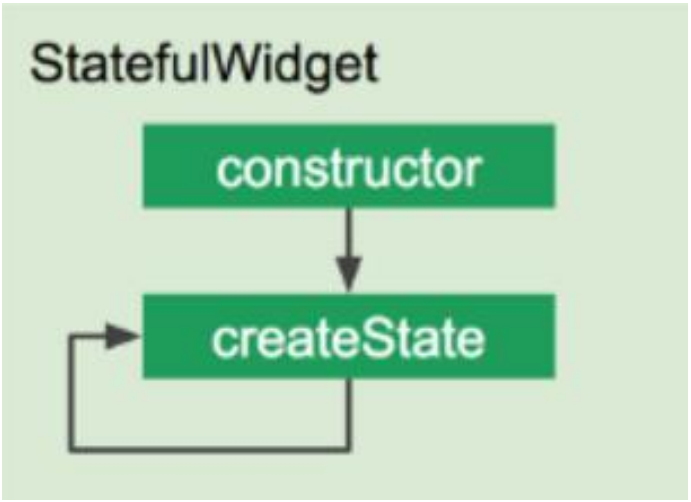


Рисунок 2.6 – Схема роботи StatefulWidget

2.3 Конструювання програмного забезпечення

В даному програмному забезпеченні існує три види об’єктів, а саме:

– сервіси;

– моделі;

– об’єкти графічного інтерфейсу.

Сервіси – це класи, що призначені для виконання певної логіки. Вони не містять ніяких даних, а лише надають послуги. В застосунку існує три сервіси, а саме:

– MethodService – призначений для надсилання запитів GET, POST, PUT, DELETE на віддалений сервер;

– ParameterCreator – призначений для створення графічних елементів в залежності від вхідних параметрів;

– SwaggerService – призначений для отримання метаданих.

Специфікація методів класу MethodService описана в таблиці 2.1.

Таблиця 2.1 – Специфікація методів класу MethodService

Назва	Аргументи	Значення, яке повертається	Призначення методу
sendGetRequest	path, headers	SwaggerResponse	Надсилає запит типу GET на адресу зі шляхом path, вносячи headers в заголовок запиту та повертає оброблену відповідь
sendPostRequest	path, headers, body	SwaggerResponse	Надсилає запит типу POST на адресу зі шляхом path, вносячи headers в заголовок, а body в тіло

Продовження таблиці 2.1

			повідомлення та повертає оброблену відповідь
Назва	Аргументи	Значення, яке повертається	Призначення методу
sendPutRequest	path, headers, body	SwaggerResponse	Надсилає PUT запит, та повертає оброблену відповідь
sendDeleteRequest	path, headers	SwaggerResponse	Надсилає DELETE запит, та повертає оброблену відповідь

Специфікація методів класу ParameterCreator описана в таблиці 2.2.

Таблиця 2.2 – Специфікація методів класу ParameterCreator

Назва	Аргументи	Значення, яке повертається	Призначення методу
generateParameterUI	ParameterCreationModel	Widget	Приймає модель даних, на основі яких вибирає відповідний віджет та повертає його
_createString	ParameterCreationModel	Widget	Повертає текстове поле для вводу, або календар, у випадку, якщо формат моделі – дата
_createNumber	ParameterCreationModel	Widget	Повертає текстове поле з валідацією на те, чи введений текст є числом,

Продовження таблиці 2.2

			або ж випадаючий список, у випадку, якщо формат даних – enum
Назва	Аргументи	Значення, яке повертається	Призначення методу
_createBoolean	ParameterCreationModel	Widget	Створює прапорець
_createArray	ParameterCreationModel	Widget	Створює спеціальний віджет, що описує масив значень
_createObject	ParameterCreationModel	Widget	Створює спеціальний віджет, що описує певний складний об’єкт

Специфікація методів класу SwaggerService описана в таблиці 2.3.

Таблиця 2.3 – Специфікація методів SwaggerService

Назва	Аргументи	Значення, яке повертається	Призначення методу
getSwaggerModel	url	SwaggerModel	Отримує необхідні метадані та створює на їх основі об’єкт
getControllerUIModel	SwaggerModel	ControllerUIModel[]	На основі створеного об’єкту,

Продовження таблиці 2.3

			формує моделі для UI частини

Моделі – це максимально прості класи, що містять лише набір даних. В застосунку існує два види моделей:

– моделі, що являють собою оброблені метадані;

– моделі, що використовуються в побудові графічного інтерфейсу.

Кожна модель, що описує метадані, містить іменований конструктор fromJSON для полегшення створення даного об'єкту. UML-діаграму моделей зображено на рисунках 2.6, 2.7.

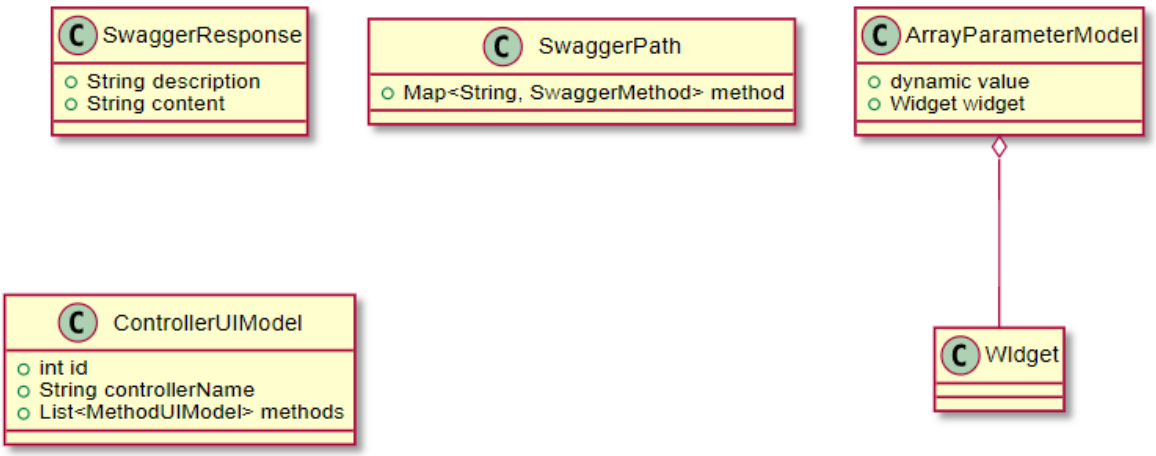


Рисунок 2.7 – Схема структурна класів моделі

Рисунок 2.8 – Схема структурна класів моделі

Віджети потрібні для створення графічного інтерфейсу. Функціональне призначення даних класів представлено в таблиці 2.4.

Таблиця 2.4 – Функціональне призначення віджетів

Назва	Тип	Функціональне призначення
MyApp	Stateless	Корінний віджет, з якого починається програма. Містить глобальну інформацію про додаток, таку як: кольорова тема, заголовок і так далі.
WelcomePage	Stateless	Початкова сторінка, що містить віджет Welcome
ControllersPage	Stateless	Друга сторінка, що призначена для відображення списку контролерів
ErrorText	Stateless	Відображує повідомлення про помилку, надаючи йому відповідний стиль
Welcome	Stateless	Відображує поле для вводу адреси необхідного сервісу, та кнопку «Execute»
NonScrollableList	Stateless	Являє собою список значень, які не можна прокручувати. Даний віджет є ефективним, адже будує лише ті елементи, які користувач може бачити в даний момент.
ControllerList	Stateless	Зображує список усіх контролерів
Controller	Stateful	Відображає список усіх методів даного контролера
MethodList	Stateless	Зображує список методів
Method	Stateful	Відображає MethodDetails, надаючи йому додаткові стилі

Продовження таблиці 2.4

Назва	Тип	Функціональне призначення
MethodDetails	Stateless	Відображає детальну інформацію про метод. А саме: секції parameters, request body, responses
Parameters	Stateless	Відображає список усіх параметрів, що повинні надходити з голови (header) запиту
RequestBody	Stateless	Відображає список усіх параметрів, що повинні надходити з тіла запиту.
Responses	Stateful	Відображає можливу та отриману відповідь, а також процес чекання на закінчення процесу.
ArrayParameter	Stateful	Надає функціональність графічного списку. Користувач має змогу додати новий елемент, внести певні дані в нього і видалити потрібний елемент
CustomCheckbox	Stateful	Реалізує функціональність прапорця, додавши додатковий стиль
CustomDateTimePicker	Stateful	Надає можливість відкрити вбудований в ОС календар для обрання дати та часу
CustomTextFormField	Stateless	Надає можливість вводити текстові дані у відповідне поле, при цьому додавши додатковий стиль до цього поля
Header	Stateless	Відображення заголовків
ObjectParameter	Stateless	Відображення графічного представлення складного об’єкта

Продовження таблиці 2.4

Назва	Тип	Функціональне призначення
ParametersGroup	Stateless	Графічний декоратор для параметрів
ParametersTemplate	Stateless	Шаблон для створення параметрів
AuthorizeButton	Stateful	Кнопка, що призначена для задання токена авторизації
AuthorizeDialog	Stateless	Модальне вікно, в якому можна отримати інформацію про токен авторизації, та ввести його у відповідне поле і зберегти.

2.4 Аналіз безпеки даних

В даному програмному забезпеченні не проводиться зберігання даних. Тобто при повторному відкритті додатку, користувачу доведеться знову вводити адресу сервера, та заповнювати параметри методу. Сервіс має змогу використовувати не тільки HTTP, але й HTTPS протокол. Проте, це не залежить від самого додатку, те , яким протоколом слід користуватись. Адже все залежить від користувача та від сервісу, з яким він вирішив спілкуватись.

Отже, програмне забезпечення є безпечним до тих пір, поки використовуються захищені протоколи, але це вже залежить від самого користувача.

2.5 Висновки по розділу

В даному розділі було виконано моделювання та аналіз мобільного застосунку. Проілюстровано бізнес-процеси з використанням діаграм BPMN. Визначено архітектуру програмного забезпечення, описано класи та їхнє функціональне призначення.

Проаналізувавши безпеку даних, зроблено висновок, що додаток є безпечним до тих пір, поки користувач використовує захищені протоколи для спілкування.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Якість програмного забезпечення – це ступінь відповідності ПЗ заданим вимогам або очікуванням користувача. Для того, щоб визначити якість системи, використовують атрибути якості (характеристики). До таких атрибутів відносять:

- надійність (reliability);
- готовність (availability);
- безпека (safety);
- зручність роботи (usability);
- ремонтопридатність (maintainability);
- конфіденційність (confidentiality);
- цілісність (integrity).

Тестування мобільного застосунку дещо відрізняється від звичайної програми для ПК. Адже потрібно враховувати мобільність пристрою, можливі оновлення операційної системи, змінення орієнтації екрану, переключення програми в фоновий режим, обмежена кількість пам'яті як основної так і оперативної і так далі.

Оскільки, метою даного розділу є опис процесу тестування мобільного додатку для автоматизованого створення HTTP запитів, надзвичайно важливим є перевірка працездатності застосунку у випадку, коли виникає проблема із зв’язком, чи то через відсутність Інтернет з’єднання, чи через проблеми на стороні сервера.

3.2 Опис процесів тестування

Тестування програмного забезпечення буде проведено вручну, а саме самим розробником даного застосунку. Об’єктом тестування виступає

мобільний додаток, який повинен автоматизовувати процес створення HTTP запитів.

Для того, щоб застосунок поводив себе коректно необхідно провести наступні тестування:

– тестування зручності у користуванні. Чи інтуїтивно зрозуміло користувачу, що потрібно зробити, для того, щоб досягти своєї мети;

– тестування інтерфейсу користувача. Зокрема, кольорова тема не повинна приносити дискомфорт, а навпаки має бути приємною оку;

– тестування зміни орієнтації екрану. В мобільних застосунках, на відміну від десктопних, можна міняти орієнтацію екрану. Через це нерідко виникають помилки;

– тестування поведінки ПЗ у випадку відсутності інтернет з’єднання;

– тестування обробки вхідних метаданих. Чи можливо їх узагалі обробити. Програма не повинна ламатись, коли спробує обробити неправильні дані;

– тестування правильності побудови графічних об’єктів;

– тестування поведінки програми, коли було створено велику кількість метаданих. Додаток повинен передбачити випадок утворення такої кількості контролерів, що не вміщується в розміри екрану. В такому разі, повинна бути можливість прокручувати список до потрібного об’єкту;

– тестування продуктивності та швидкодії ПЗ у випадку великої кількості даних;

– тестування розгортання контролера при кліку, та відображення списку методів;

– тестування розгортання методу при кліку та відображення списку параметрів даного методу;

– тестування вводу даних у відповідні поля;

– тестування правильності побудови графічних об’єктів на основі їх метаданих;

– тестування валідації даних. Так, наприклад, якщо тип об’єкту – текст, то при спробі виконати перевірку, не повинна виникнути помилка про те, що даний об’єкт мусить містити тільки число;

– тестування надсилання даних та отримання відповіді від сервера.

3.3 Опис контрольного прикладу

Опис контрольних прикладів наведено в таблицях 3.1 – 3.12.

Таблиця 3.1 – Перевірка надсилання запиту

Мета тесту	Перевірка надсилання запиту для отримання метаданих
Передумови	Користувач відкрив початкову сторінку
Схема виконання	Користувач вводить правильну адресу у відповідне поле
Очікуваний результат	Метадані успішно отримано
Дійсний результат	Метадані успішно отримано

Таблиця 3.2 – Перевірка оброблення помилки при виконанні запиту

Мета тесту	Перевірка оброблення помилки при виконанні запиту для отримання метаданих
Передумови	Користувач відкрив початкову сторінку
Схема виконання	Користувач вводить неправильну адресу у відповідне поле
Очікуваний результат	Повідомлення про помилку було відображено у відповідному полі
Дійсний результат	Повідомлення про помилку було відображено у відповідному полі

Таблиця 3.3 – Перевірка правильності побудови графічних об’єктів

Мета тесту	Перевірка правильності побудови графічних об’єктів
Передумови	Користувач відкрив початкову сторінку, та вже знає, усю інформацію про можливі отримані дані. Тобто кількість та назву контролерів, методів та їх параметрів. Наприклад, кількість контролерів повинна дорівнювати 5
Схема виконання	Користувач вводить адресу у відповідне поле та натискає на кнопку підтвердження Застосунок обробляє дані та відкриває сторінку зі списком контролерів
Очікуваний результат	Сторінка зі списком контролерів успішно відкрита, на якій зображено 5 контролерів
Дійсний результат	Сторінка зі списком контролерів успішно відкрита, на якій зображено 5 контролерів

Таблиця 3.4 – Перевірка можливості розгорнути контролера

Мета тесту	Перевірка можливості розгорнути контролер
Передумови	Метадані успішно оброблені і побудовані графічні об’єкти
Схема виконання	Користувач натискає на контролер
Очікуваний результат	Додаток реагує на подію нажимання кнопки та відображує список методів даного контролера
Дійсний результат	Додаток реагує на подію нажимання кнопки та відображує список методів даного контролера

Таблиця 3.5 – Перевірка можливості розгорнути метод

Мета тесту	Перевірка можливості розгорнути метод
Передумови	Метадані успішно оброблені і побудовані графічні об’єкти Певний контролер розгорнутий, тож користувачу відображається список його методів
Схема виконання	Користувач натискає на певний метод
Очікуваний результат	Додаток реагує на подію нажимання кнопки та відображує список усіх параметрів даного методу
Дійсний результат	Додаток реагує на подію нажимання кнопки та відображує список усіх параметрів даного методу

Таблиця 3.6 – Перевірка можливості розгорнути метод

Мета тесту	Перевірка можливості розгорнути метод
Передумови	Метадані успішно оброблені і побудовані графічні об’єкти Певний контролер розгорнутий, тож користувачу відображається список його методів
Схема виконання	Користувач натискає на певний метод
Очікуваний результат	Додаток реагує на подію нажимання кнопки та відображує список усіх параметрів даного методу
Дійсний результат	Додаток реагує на подію нажимання кнопки та відображує список усіх параметрів даного методу

Таблиця 3.7 – Перевірка правильності побудови графічних параметрів

Мета тесту	Перевірка правильності побудови графічних параметрів у залежності від типу та формату

Продовження таблиці 3.7

Передумови	Певний метод «А» містить два параметри даних типів: текстовий, дата. Користувач розгорнув контролер, у якому міститься даний метод «А»
Схема виконання	Користувач натискає на метод «А»
Очікуваний результат	Додаток відображає 2 графічні параметри. Перший – це текстове поле для вводу, другий – кнопка, що відкриває вбудований в ОС календар
Дійсний результат	Додаток відображає 2 графічні параметри. Перший – це текстове поле для вводу, другий – кнопка, що відкриває вбудований в ОС календар

Таблиця 3.8 – Перевірка правильності валідації параметрів

Мета тесту	Перевірка правильності валідації параметрів
Передумови	Певний метод містить один параметр з форматом GUID Користувач розгорнув метод «А»
Схема виконання	Користувач вводить значення, що не є форматом GUID в параметр даного методу Користувач натискає на кнопку виконання запиту
Очікуваний результат	Застосунок проводить валідацію даних. Оскільки введені дані не є правильними, відображається повідомлення про невалідність під відповідним параметром
Дійсний результат	Застосунок проводить валідацію даних. Оскільки введені дані не є правильними, відображається відповідне повідомлення про невалідність

Таблиця 3.9 – Перевірка надсилання запиту

Мета тесту	Перевірка надсилання запиту
Передумови	Користувач ввів необхідні та правильні дані у відповідні поля параметрів
Схема виконання	Користувач натискає на кнопку виконання запиту
Очікуваний результат	Запит успішно надсилається, приходить відповідь, яка відображується у відповідній секції
Дійсний результат	Запит успішно надсилається, приходить відповідь, яка відображується у відповідній секції

Таблиця 3.10 – Перевірка працездатності при зміні орієнтації

Мета тесту	Перевірка працездатності програми при зміні орієнтації екрану
Передумови	Користувач відкрив програму в портретному режимі
Схема виконання	Користувач перевертає свій мобільний девайс з метою переключити програму в альбомний режим
Очікуваний результат	Програма успішно міняє орієнтацію, при цьому не ламаючись
Дійсний результат	Програма успішно міняє орієнтацію, при цьому не ламаючись

Таблиця 3.11 – Перевірка відображення кнопки авторизації

Мета тесту	Перевірка відображення кнопки авторизації
Передумови	Для того, щоб виконувати запити на сервер необхідно бути авторизованим
Схема виконання	Користувач відриває сторінку «Controllers»

Продовження таблиці 3.11

Очікуваний результат	Оскільки для виконання запиту необхідна авторизація, повинна бути відображена кнопка авторизації
Дійсний результат	Оскільки для виконання запиту необхідна авторизація, повинна бути відображена кнопка авторизації

Таблиця 3.12 – Перевірка задання токена авторизації

Мета тесту	Перевірка задання токена авторизації
Передумови	Користувач відкрив сторінку авторизації. Та для виконання певного запиту необхідна авторизація
Схема виконання	Користувач натискає на кнопку «Authorize». Після цього буде відображено модальне вікно, яке містить поле для вводу, короткий опис про токен та кнопки підтвердження і скасування операції. Користувач вводить токен в поле для вводу та натискає на кнопку підтвердження
Очікуваний результат	Токен авторизації повинен бути збереженим. Модальне вікно має заховатись, а кнопка авторизації повинна змінити свій колір та ікнонку
Дійсний результат	Токен авторизації повинен бути збереженим. Модальне вікно має заховатись, а кнопка авторизації повинна змінити свій колір та ікнонку

3.4 Висновки по розділу

В даному розділі було проведено аналіз якості програмного забезпечення. Для цього перш за все необхідно було виявнити, що узагалі представляє собою якість ПЗ, які вона має характеристики, атрибути. Було визначено ряд тестів, які застосунок повинен пройти для того, щоб ним можна було користуватись. Було проведено дані тести, які мобільний застосунок успішно пройшов. Тож можна зробити висновок, що даний додаток є цілком працездатним та відмовостійким. А також має зручний інтерфейс для комфортного користування.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для того, щоб розпочати роботу з програмним застосунком, необхідна наявність мобільного приладу з операційною системою на базі «Android». Версія даної ОС повинна бути не нижче 6.0. Необхідно завантажити apk-файл додатку на мобільний девайс. Після цього потрібно виконати встановлення даного додатку. Більшість операційних систем, що використовують Android в якості ядра, мають вбудований інсталятор, що здатен встановлювати apk-файли. Проте у разі, якщо дана ОС такого не підтримує, доведеться додатково завантажити додаток для встановлення таких файлів. Це можна зробити з офіційного магазину «Google Play Store». Після успішного встановлення, застосунок можна запустити, клацнувши на його іконку.

4.2 Робота з програмним забезпеченням

Детальну інформацію по роботі з даним мобільним застосунком наведено в додатку «Керівництво користувача».

4.3 Висновки по розділу

У даному розділі було визначено і описано процес розгортання програмного забезпечення. Також продемонстровано процес роботи з мобільним застосунком.

ВИСНОВКИ

Протягом виконання даної дипломної роботи було проаналізовано проблему створення HTTP API запитів з мобільних платформ. Дослідження аналогів показало, що на сьогоднішній момент на ринку не існує додатків, здатних дозволити користувачам зручним чином виконувати необхідні запити на веб-сервер.

Було спроектовано і написано мобільний застосунок для операційних систем, що використовують “Android” у якості ядра. Даний додаток має ряд переваг перед своїми аналогами. Серед них можна виділити наступні:

– додаток надає необхідну інформацію для створення HTTP API запитів на відповідний сервер, адже створює графічне відображення списку усіх контролерів, їхніх методів та параметрів усередині даних методів;

– додаток надає можливість проводити валідацію параметрів метода перш ніж виконати запит. Таким чином, зменшуючи ризики у створенні неправильного повідомлення;

– застосунок надає можливість задати токен авторизації там, де це потрібно. Тобто користувач завжди буде знати: чи необхідно авторизовуватись для виконання певного запиту, чи ні;

– користувач має змогу вводити дані відповідно від їхнього типу. Тобто, якщо тип - дата, то він не буде вручну вводити дані, адже це не зручно. Натомість - використає вбудований в ОС календар;

– графічний інтерфейс є сучасним і зручним у використанні.

У ході розробки програмного забезпечення також було проведено тестування, яке показало ряд недоліків і дозволило їх усунути.

У результаті створення даної роботи було розроблено документацію проекту, побудовано діаграми зв’язків, створені таблиці, що описують функціонал, варіанти використання, а також було написано керівництво користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1) Flutter документація [Електронний ресурс]: (Стаття) / Flutter. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://flutter.dev/docs. – Назва з екрана.

2) HTTP [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://uk.wikipedia.org/wiki/HTTP. – Назва з екрана.

3) Прикладний програмний інтерфейс [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://en.wikipedia.org/wiki/Application_programming_interface. – Назва з екрана.

4) Клієнт серверна архітектура [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://en.wikipedia.org/wiki/Client%E2%80%93server_model. – Назва з екрана.

5) Прикладний програмний інтерфейс [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://en.wikipedia.org/wiki/Application_programming_interface. – Назва з екрана.

6) Android SDK [Електронний ресурс]: (Стаття) / Android Studio. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://developer.android.com/studio/releases/sdk-tools. – Назва з екрана.

7) REST [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://uk.wikipedia.org/wiki/REST. – Назва з екрана.

8) Інтернет [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://en.wikipedia.org/wiki/Internet. – Назва з екрана.

9) Токен автентифікації [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://en.wikipedia.org/wiki/Security_token. – Назва з екрана.

10) Open API [Електронний ресурс]: (Стаття) / Open API. – Електрон. дан. (1 файл). – 2020 – Режим доступу: https://www.openapis.org/. – Назва з екрана.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Технічне завдання

КПІ.ІП-6314.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ Нечай Д.О.

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ Карпа М. В.

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: мобільне застосування для автоматизації роботи з HTTP API на основі метаданих.

Галузь застосування: будь-яка сфера, де необхідно виконувати HTTP API запити.

Наведене технічне завдання поширюється на розробку мобільного застосунку «Swagger Mobile Client» [SMC], котра використовується для автоматизації та полегшення роботи з HTTP API та призначена для будь-кого, кому необхідно надсилати запити на віддалений сервер.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки «Swagger Mobile Client» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та полегшенню роботи з HTTP API на мобільних пристроях.

Метою розробки є підвищення ефективності та спрощення виконання HTTP запитів з мобільного пристрою, мінімізація помилок при формуванні відповідного запиту.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1.1 Для користувача:

– можливість вводити url-адресу необхідного сервісу;

– динамічна побудова графічних об’єктів на основі метаданих;

– можливість розгортати контролери, для отримання інформації про їхні методи;

– можливість розгортати методи, для отримання інформації про їхні параметри;

– можливість вводити дані в параметри методів;

– валідація введених даних;

– надсилання введених даних на відповідну кінцеву точку.

4.1.2 Розробку виконати на платформі Android

4.1.3 Додаткові вимоги:

– для успішної роботи програми, сервер, який буде тестуватись повинен реалізовувати інтерфейс OpenApi 3.0.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються.

4.3.2 Обслуговування та обслуговуючий персонал.

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на мобільних пристроях з операційною системою Android.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору

Будь-який процесор, що використовується на сучасних смартфонах.

4.4.2.2 Об‘єм ОЗП

500 Мб, або більше.

4.5 Вимоги до інформаційної та програмної сумісності

– програмне забезпечення повинно працювати під управлінням операційних систем сімейства Android;

– вхідні дані повинні бути представлені в наступному форматі: текстові, числові, дата та час, булеві;

– результати повинні бути представлені в наступному форматі: текст формату JSON;

– програмне забезпечення повинно вміти обробляти метадані, що надходять від сервера, який реалізує інтерфейс OpenApi 3.0.

Мобільний застосунок був розроблений за допомогою програмного каркасу Flutter. Основною мовою програмування є Dart. Середовище розробки було вибрано Visual Studio Code.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати установочну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації.

а) Супроводжувальна документація:

1) пояснювальна записка;

2) керівництво користувача;

3) програма та методика тестування.

б) Довідникова документація:

1) програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі;

2) програмне забезпечення повинно мати вбудовану довідку.

в) Графічна документація:

1) схема структурна варіантів використання;

2) схема структурна діяльності;

3) схема структурна класів програмного забезпечення.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02.2020	
2.	Розробка технічного завдання	03.03.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	20.04.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування програмного забезпечення описано в додатку «Програма та методика тестування».

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Опис програми

КПІ.ІП-6314.045440.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Д.О. Нечай

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М.В. Карпа

Київ – 2020 року

Тексти програмного коду

Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих

(Найменування програми (документа))

DVD-R

(Вид носія даних)

46арк, 392 Мб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Опис програми

КПІ.ІП-6314.045440.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Д.О. Нечай

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М.В. Карпа

Київ – 2020 року

Тексти програмного коду

Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих

(Найменування програми (документа))

DVD-R

(Вид носія даних)

46арк, 392 Мб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

Файл method_service.dart

```
import 'dart:convert';

import 'package:http/http.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/models/swagger_ui_model.dart';

// Used for executing GET, POST, PUT, DELETE methods
class MethodService {
  final String url;

  MethodService(this.url);

  Future<Map<String, SwaggerResponse>> sendRequest({
    MethodUIModel model,
    Map<String, dynamic> parameters,
    Map<String, dynamic> requestBody,
    String authorizationValue
  }) {
    final path = _getPathWithParameters(model.endPoint, parameters);
    final headers = _getHeaders(authorizationValue);

    switch (model.type) {
      case "get":
        return sendGetRequest(path, headers);
      case "post":
        return sendPostRequest(path, headers, requestBody);
      case "put":
        return sendPutRequest(path, headers, requestBody);
      case "delete":
        return sendDeleteRequest(path, headers);
    }
    return null;
  }

  /* Public methods */
  Future<Map<String, SwaggerResponse>> sendGetRequest(
    String path,
    Map<String, String> headers,
  ) async {
    final response = await get(path, headers: headers);
    return _getSwaggerResponse(response);
  }

  Future<Map<String, SwaggerResponse>> sendPostRequest(
    String path,
```

```
    Map<String, String> headers,
    Map<String, dynamic> requestBody,
  ) async {
    final body = json.encode(requestBody);
    print(body);
    final response = await post(
      path,
      headers: headers,
      body: json.encode(requestBody),
    );
    return _getSwaggerResponse(response);
  }

  Future<Map<String, SwaggerResponse>> sendPutRequest(
    String path,
    Map<String, String> headers,
    Map<String, dynamic> requestBody,
  ) async {
    final body = json.encode(requestBody);
    print(body);
    final response = await put(
      path,
      headers: headers,
      body: body,
    );
    return _getSwaggerResponse(response);
  }

  Future<Map<String, SwaggerResponse>> sendDeleteRequest(
    String path,
    Map<String, String> headers,
  ) async {
    final response = await delete(path, headers: headers);
    return _getSwaggerResponse(response);
  }

  /* Helpful private methods */
  String _getPathWithParameters(endpoint, parameters) {
    var result = "$url$endpoint";
    parameters.forEach((key, value) {
      result = result.replaceAll('{$key}', value);
    });
    return result;
  }

  Map<String, SwaggerResponse> _getSwaggerResponse(Response response) {
    return Map.fromEntries([
      MapEntry(
```

```
        response.statusCode.toString(),
        SwaggerResponse(
          description: response.reasonPhrase,
          content: response.bodyBytes.length == 0
              ? ""
              : JsonEncoder.withIndent('  ').convert(
                  json.decode(response.body),
                ),
        ),
      ),
    ]);
  }

  Map<String, String> _getHeaders(String authorizationValue) {
    final headers = {
      "content-type": "application/json",
      "accept": "application/json",
    };

    if(authorizationValue != null && authorizationValue.isNotEmpty) {
      headers["authorization"] = authorizationValue;
    }

    return headers;
  }
}
```

Файл parameter_creator.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/parameter_creation_model.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/services/validator_service.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/custom_datepicker_w
idget.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/custom_text_form_fi
eld_widget.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/array_parameter_wid
get.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/object_parameter_wi
dget.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/custom_checkbox_wid
get.dart';

// TODO: Refactor this
// TODO: by adding supportigng all formats and types
```

```dart
class ParameterCreator {
  /* Singleton */
  static final ParameterCreator _singleton = ParameterCreator._internal();
  factory ParameterCreator() => _singleton;
  ParameterCreator._internal();

  final _validatorService = ValidatorService();

  /* Methods */
  Widget generateParameterUI(ParameterCreationModel model) {
    if (model == null) return SizedBox.shrink();

    Widget widget;
    switch (model?.schema?.type) {
      case "string":
        widget = _createString(model);
        break;
      case "number":
      case "integer":
        widget = _createNumber(model);
        break;
      case "boolean":
        widget = _createBoolean(model);
        break;
      case "array":
        widget = _createArray(model);
        break;
      case "object":
        widget = _createObject(model);
        break;
    }
    return Container(
      child: widget,
      margin: EdgeInsets.only(bottom: 5),
    );
  }

  Map<String, dynamic> initValuesMap(List<SwaggerParameterSchema> parameters) {
    final Map<String, dynamic> result = Map();
    parameters?.forEach((p) {
      var value;
      switch (p?.type) {
        case "array":
          value = [];
          break;
        case "boolean":
          value = false;
          break;
```

```
        case "object":
          value = initValuesMap(p.childSchemas);
          break;
        default:
          value = null;
          break;
      }
      p.name == "" && value is Map
          ? result.addAll(value)
          : result.addEntries([MapEntry(p.name, value)]);
    });

    return result;
  }

  /* Helpful methods */

  /// Official formats:
  /// type: string, no format
  /// type: string, format - byte (base64 encoded characters)
  /// type: string, format - binary (any sequence of octets)
  /// type: string, format - date (date-fullyear "-" date-month "-" date-mday)
  /// type: string, format - date-time (full-date "T" full-time)
  ///
  /// Non-official formats:
  /// type: string, format - uuid (guid)
  /// type: string, format - email
  Widget _createString(ParameterCreationModel model) {
    switch (model.schema.format) {
      case "date-time":
        return CustomDateTimePicker(
          name: _getName(model.name, model.schema.name),
          validator: (value) => _validatorService.validateDate(
            value,
            model.schema.isRequired,
          ),
          onChanged: (v) => model.onChangedHandler(
            _getName(
              model.name,
              model.schema.name,
            ),
            v.toIso8601String(),
          ),
        );
      default:
        return CustomTextFormField(
          label: model.schema.name,
          textInputType: TextInputType.text,
```

```
          validator: (value) => _validatorService.validateString(
            value,
            model.schema.format,
            model.schema.isRequired,
          ),
          onChanged: (v) => model.onChangedHandler(
            _getName(
              model.name,
              model.schema.name,
            ),
            v,
          ),
        );
    }
  }

  /// type - integer, format - int32
  /// type - integer, format - int64
  /// type - number, format - float
  /// type - number, format - double
  Widget _createNumber(ParameterCreationModel model) {
    if (model.schema.enumTypes != null) {
      return _createEnum(model);
    } else {
      return CustomTextFormField(
        label: model.schema.name,
        textInputType: TextInputType.number,
        validator: (value) => _validatorService.validateNumber(
          value,
          model.schema.format,
          model.schema.isRequired,
        ),
        onChanged: (v) => model.onChangedHandler(
          _getName(
            model.name,
            model.schema.name,
          ),
          v,
        ),
      );
    }
  }

  Widget _createEnum(ParameterCreationModel model) {
    return DropdownButtonFormField(
      items: model.schema.enumTypes.map((dynamic value) {
        return DropdownMenuItem(
          value: value,
```

```
          child: Text(value.toString()),
        );
      }).toList(),
      validator: (value) => _validatorService.validateEnum(
        value,
        model.schema.isRequired,
      ),
      onChanged: (v) => model.onChangedHandler(
        _getName(
          model.name,
          model.schema.name,
        ),
        v,
      ),
      decoration: InputDecoration(
        labelText: '${model.schema.name}',
        hintText: '${model.schema.name}',
        labelStyle: TextStyle(
          decorationStyle: TextDecorationStyle.solid,
        ),
      ),
    );
  }

  /// type - boolean, no format
  Widget _createBoolean(ParameterCreationModel model) {
    return CustomCheckbox(
      name: model.schema.name,
      onChanged: (v) => model.onChangedHandler(
        _getName(
          model.name,
          model.schema.name,
        ),
        v,
      ),
    );
  }

  Widget _createArray(ParameterCreationModel model) {
    return ArrayParameter(
      parameter: model.schema,
      onChanged: (v) =>
          model.onChangedHandler(_getName(model.name, model.schema.name), v),
    );
  }

  Widget _createObject(ParameterCreationModel model) {
    return ObjectParameter(
```

```
        parameter: model.schema,
        onChanged: (name, value) => model.onChangedHandler(name, value),
      );
    }

    String _getName(String modelName, String schemaName) {
      return modelName == null ? schemaName : modelName;
    }
  }
```

Файл swagger_service.dart

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart';

import 'package:swagger_mobile_client/models/swagger_json/swagger_model.dart';
import 'package:swagger_mobile_client/models/swagger_ui_model.dart';

class SwaggerService {
  /* Singleton */
  static final SwaggerService _singleton = SwaggerService._internal();
  factory SwaggerService() => _singleton;
  SwaggerService._internal();

  Future<SwaggerModel> getSwaggerModel(url, swaggerEndpoint) async {
    final response = await get(_getUri(url, swaggerEndpoint));
    final text = json.decode(response.body);
    return SwaggerModel.fromJSON(url, text);
  }

  List<ControllerUIModel> getControllerUIModel(SwaggerModel model) {
    var methodUIModels = List<MethodUIModel>();
    model.paths.entries.forEach((pathEntry) {
      pathEntry.value.method.entries.forEach((methodEntry) {
        methodUIModels.add(
          MethodUIModel(
            controllerName: methodEntry.value.tags[0],
            endPoint: pathEntry.key,
            type: methodEntry.key,
            detail: MethodUIDetailModel(
              parameters: methodEntry.value.parameters,
              requestBody: methodEntry.value.requestBody,
              responses: ValueNotifier(methodEntry.value.responses),
            ),
          ),
        );
      });
```

```
    });

    final grouped = _groupBy<MethodUIModel, String>(
        methodUIModels, (m) => m.controllerName);
    final result = grouped.entries.map<ControllerUIModel>((entry) {
      return ControllerUIModel(
        controllerName: entry.key,
        methods: entry.value,
      );
    }).toList();

    return result;
  }

  Map<T, List<S>> _groupBy<S, T>(Iterable<S> values, T key(S element)) {
    var map = <T, List<S>>{};
    for (var element in values) {
      var list = map.putIfAbsent(key(element), () => []);
      list.add(element);
    }
    return map;
  }

  String _getUri(String url, String swaggerEndpoint) {
    return "$url/$swaggerEndpoint";
  }

}
```

Файл main.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/route_consts.dart';
import 'package:swagger_mobile_client/widgets/pages/welcome_page.dart';

import 'models/consts/string_consts.dart';
import 'widgets/pages/controllers_page.dart';
import 'widgets/pages/welcome_page.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: StringConsts.appName,
      theme: ThemeData(
        brightness: Brightness.light,
```

```
        primarySwatch: Colors.lightGreen,
        buttonTheme: ButtonThemeData(
          colorScheme: ColorScheme.fromSwatch(primarySwatch: Colors.blue),
          textTheme: ButtonTextTheme.primary,
        ),
      ),

      // Routes
      initialRoute: RouteConsts.welcomePage,
      routes: {
        RouteConsts.welcomePage : (context) => WelcomePage(),
        RouteConsts.controllersPage : (context) => ControllersPage(ModalRoute.of(c
ontext).settings.arguments)
      },
    );
  }
}
```

Файл route_consts.dart

```
class RouteConsts {
  static const String welcomePage = "/";
  static const String controllersPage = "/controllers";
}
```

Файл string_consts.dart

```
class StringConsts {
  /* App */
  static const String appName = "Swagger Client";

  /* Controllers Page */
  static const String controllersTitle = "Controllers";
  static const String executeBtn = "Execute";
  static const String parameters = "Parameters";
  static const String requestBody = "Request body";
  static const String responses = "Responses";
  static const String parameterName = "Name";
  static const String parameterValue = "Value";
  static const String datePickerBtn = "Pick";


  /* Welcome Page */
  static const String enterUrl = "Enter the URL";
  static const String urlLabel = "URL";
  static const String urlHint = "http://example.com";
  static const String swaggerEndpointLabel = "OpenAPI endpoint";
  static const String swaggerEndpointHint = "swagger/v1/swagger.json";
  static const String confirmBtn = "Confirm";
```

```
  static const String errorText = "An error occured!";

  /* Authorize */
  static const String authorize = "Authorize";
  static const String authorized = "Authorized";
  static const String cancelBtn = "Cancel";
  static const String okBtn = "Ok";
  static const String authorization = "Authorization";
}
```

Файл security_schemes_model.dart

```
import 'package:flutter/material.dart';

class SecuritySchemes {
  final ValueNotifier<String> authorization;
  final Map<String, SecurityScheme> schemes;

  SecuritySchemes({this.schemes, this.authorization});

  factory SecuritySchemes.fromJSON(Map<String, dynamic> json) {
    if (json == null) return null;

    return SecuritySchemes(
      schemes: json.map((key, value) {
        return MapEntry(
          key,
          SecurityScheme.fromJSON(value),
        );
      }),
      authorization: ValueNotifier("")
    );
  }
}

class SecurityScheme {
  final String type;
  final String description;
  final String name;
  final String inType;

  SecurityScheme({this.type, this.description, this.name, this.inType});

  factory SecurityScheme.fromJSON(Map<String, dynamic> json) {
    if (json == null) return null;

    return SecurityScheme(
      type: json["type"],
      description: json["description"],
```

```
      name: json["name"],
      inType: json["in"],
    );
  }
}
```

Файл swagger_method_model.dart

```
import 'dart:convert';

import 'package:swagger_mobile_client/models/consts/global_consts.dart';

class SwaggerMethod {
  final List<String> tags;
  final SwaggerParameters parameters;
  final SwaggerRequestBody requestBody;
  final Map<String, SwaggerResponse> responses;

  SwaggerMethod({this.tags, this.parameters, this.requestBody, this.responses});

  factory SwaggerMethod.fromJSON(
      Map<String, dynamic> json, Map<String, dynamic> schemas) {
    final tagsDynamic = json["tags"] as List<dynamic>;

    final requestBody = (json.containsKey("requestBody"))
        ? SwaggerRequestBody.fromJSON(json["requestBody"], schemas)
        : null;

    final responsesDynamic = json["responses"] as Map<String, dynamic>;

    return SwaggerMethod(
      tags: tagsDynamic.map((t) => t.toString()).toList(),
      parameters:
          SwaggerParameters.fromJSON(json["parameters"] as List<dynamic>),
      requestBody: requestBody,
      responses: responsesDynamic.map(
        (key, value) => MapEntry(
          key,
          SwaggerResponse.fromJSON(value, schemas),
        ),
      ),
    );
  }
}

class SwaggerParameters {
  List<SwaggerParameterSchema> content;
```

```
    SwaggerParameters({this.content});

    factory SwaggerParameters.fromJSON(List<dynamic> json) {
      if (json == null) return null;

      return SwaggerParameters(
        content: json.map((x) {
          final schemaMap = x["schema"] as Map<String, dynamic>;

          return SwaggerParameterSchema(
            name: x["name"],
            description: x["description"],
            inType: x["in"],
            isRequired: x["required"],
            format: schemaMap["format"],
            type: schemaMap["type"],
          );
        }).toList(),
      );
    }
}

class SwaggerParameterSchema {
  final String name;
  final String type;
  final String format;

  // Values of parameter
  final String inType; // path or else
  final String description;
  final bool isRequired;

  final List<dynamic> enumTypes; // in case if type = 'enum'
  final SwaggerParameterSchema arraySchema; // in case if type = 'array'
  final List<SwaggerParameterSchema> childSchemas; // in case if type = 'object'

  SwaggerParameterSchema({
    this.name,
    this.type,
    this.format,

    /* only in parameter */
    this.inType,
    this.description,
    this.isRequired,
    this.enumTypes,
    this.arraySchema,
    this.childSchemas,
```

```
  });

  factory SwaggerParameterSchema.fromJSON({
    String name,
    Map<String, dynamic> json,
    Map<String, dynamic> schemas,
    int currIteraction = 1
  }) {
    if (json == null || currIteraction > GlobalConsts.maxParameterIteraction) retu
rn null;
    var type = json["type"];
    var objects;

    if (json["\$ref"] != null) {
      final key = json["\$ref"].split('/').last;
      // enum
      if (schemas[key]["enum"] != null) {
        return SwaggerParameterSchema.fromJSON(
          name: name,
          json: schemas[key],
          schemas: schemas,
          currIteraction: currIteraction + 1
        );
      } else {
        final properties = schemas[key]["properties"] as Map<String, dynamic>;
        objects = properties?.entries?.map((entry) {
          return SwaggerParameterSchema.fromJSON(
            name: entry.key,
            json: entry.value,
            schemas: schemas,
            currIteraction: currIteraction + 1
          );
        })?.toList();

        type = schemas[key]["type"];
      }
    }
    return SwaggerParameterSchema(
      name: name,
      type: type,
      format: json["format"],
      enumTypes: json["enum"],
      arraySchema: SwaggerParameterSchema.fromJSON(
        name: name,
        json: json["items"],
        schemas: schemas,
        currIteraction: currIteraction + 1
      ),
```

```
        childSchemas: objects,
      );
    }
  }

  class SwaggerRequestBody {
    final SwaggerParameterSchema content;

    SwaggerRequestBody({this.content});

    factory SwaggerRequestBody.fromJSON(
      Map<String, dynamic> json,
      Map<String, dynamic> schemas,
    ) {
      if (json == null) return null;

      var content = SwaggerParameterSchema();
      if (json.containsKey("content")) {
        final contentMap = json["content"] as Map<String, dynamic>;
        final schemaMap = contentMap?.values?.first as Map<String, dynamic>;
        final schema = schemaMap["schema"] as Map<String, dynamic>;

        if (schema != null) {
          content = SwaggerParameterSchema.fromJSON(
            json: schema,
            schemas: schemas,
            name: '',
          );
        }
      }
      return SwaggerRequestBody(content: content);
    }
  }

  class SwaggerResponse {
    String description;
    String content;

    SwaggerResponse({this.description, this.content});

    factory SwaggerResponse.fromJSON(
        Map<String, dynamic> json, Map<String, dynamic> schemas) {
      if (json == null) return null;

      String content = "";
      if (json.containsKey("content")) {
        final contentMap = json["content"] as Map<String, dynamic>;
        final schemaMap = contentMap?.values?.first as Map<String, dynamic>;
```

```
      final refMap = schemaMap["schema"] as Map<String, dynamic>;

      if (refMap != null && refMap["\$ref"] != null) {
        final ref = refMap["\$ref"];
        final key = ref.split('/').last;
        final properties = schemas[key]["properties"];

        final contentObject = properties.map((key, value) {
          return MapEntry(key, value["type"]);
        });
        content = JsonEncoder.withIndent('  ').convert(contentObject);
      }
    }

    return SwaggerResponse(
      content: content,
      description: json["description"] as String,
    );
  }
}
```

Файл swagger_model.dart

```
import 'swagger_method_model.dart';

import 'package:swagger_mobile_client/models/swagger_json/security_schemes_model.d
art';

class SwaggerModel {
  final String url;
  final String openapi;
  final SwaggerInfo info;
  final Map<String, SwaggerPath> paths;
  final SecuritySchemes securitySchemes;

  SwaggerModel({
    this.url,
    this.openapi,
    this.info,
    this.paths,
    this.securitySchemes,
  });

  factory SwaggerModel.fromJSON(String url, Map<String, dynamic> json) {
    final pathsDynamic = json["paths"] as Map<String, dynamic>;

    return SwaggerModel(
      url: url,
```

```
      openapi: json["openapi"],
      info: SwaggerInfo.fromJSON(json["info"]),
      paths: pathsDynamic.map(
        (key, value) => MapEntry(
          key,
          SwaggerPath.fromJSON(
            value,
            json["components"],
          ),
        ),
      ),
      securitySchemes: SecuritySchemes.fromJSON(json["components"]["securityScheme
s"]),
    );
  }
}

class SwaggerInfo {
  final String title;
  final String version;

  SwaggerInfo({this.title, this.version});

  factory SwaggerInfo.fromJSON(Map<String, dynamic> json) {
    return SwaggerInfo(
      title: json["title"],
      version: json["version"],
    );
  }
}

class SwaggerPath {
  final Map<String, SwaggerMethod> method;
  SwaggerPath({this.method});

  factory SwaggerPath.fromJSON(
      Map<String, dynamic> json, Map<String, dynamic> components) {
    return SwaggerPath(
      method: json.map(
        (key, value) => MapEntry(
          key,
          SwaggerMethod.fromJSON(
            value,
            components["schemas"],
          ),
        ),
      ),
    );
```

```
  }
}
```

Файл array_parameter_model.dart

```
import 'package:flutter/material.dart';

class ArrayParameterModel {
  dynamic value;
  Widget widget;

  ArrayParameterModel({this.value, this.widget});
}
```

Файл parameter_creation_model.dart

```
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';

class ParameterCreationModel {
  final String name;
  final SwaggerParameterSchema schema;
  final Function(String, dynamic) onChangedHandler;

  ParameterCreationModel({this.name, this.schema, this.onChangedHandler});
}
```

Файл swagger_ui_model.dart

```
import 'package:flutter/material.dart';

import 'swagger_json/swagger_method_model.dart';

class ControllerUIModel
{
  final int id;
  final String controllerName;
  final List<MethodUIModel> methods;

  ControllerUIModel({this.id, this.controllerName, this.methods});
}

class MethodUIModel
{
  final int id;
  final String controllerName;
  final String endPoint;
  final String type;
  final MethodUIDetailModel detail;

  Color get typeColor {
```

```
    if(type == 'get') return Colors.green;
    if(type == 'post') return Colors.blue;
    if(type == 'put') return Colors.orange;
    if(type == 'delete') return Colors.red;

    return Colors.black;
  }

  MethodUIModel({this.id, this.controllerName, this.endPoint, this.type, this.deta
il});
}

class MethodUIDetailModel {
  SwaggerParameters parameters;
  SwaggerRequestBody requestBody;
  ValueNotifier<Map<String, SwaggerResponse>> responses;

  MethodUIDetailModel({this.parameters, this.requestBody, this.responses});
}
```

Файл non_scrollable_list_widget.dart

```
import 'package:flutter/material.dart';

class NonScrollableList extends StatelessWidget {
  final Function(BuildContext,int) itemBuilder;
  final int itemCount;

  NonScrollableList({this.itemBuilder, this.itemCount});

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      padding: const EdgeInsets.all(5),
      itemCount: itemCount,
      itemBuilder: itemBuilder,
      scrollDirection: Axis.vertical,
      shrinkWrap: true,
      physics: const NeverScrollableScrollPhysics()
    );
  }
}
```

Файл controllers_page.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_model.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/authorize/authorize
_button_widget.dart';
```

```
import 'package:swagger_mobile_client/widgets/controllers_page/controller_list_wid
get.dart';

class ControllersPage extends StatelessWidget {
  final SwaggerModel _swaggerModel;
  ControllersPage(this._swaggerModel);

  List<Widget> getActions() {
    return (_swaggerModel.securitySchemes != null)
        ? <Widget>[
            Container(
              padding: EdgeInsets.only(right: 8),
              child: AuthorizeButton(
                _swaggerModel.securitySchemes
              ),
            ),
          ]
        : [];
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: const Text(StringConsts.controllersTitle),
          actions: getActions()),
      body: Container(
        child: SingleChildScrollView(
          child: ControllerList(_swaggerModel),
        ),
      ),
    );
  }
}
```

Файл welcome_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';
import 'package:swagger_mobile_client/models/consts/route_consts.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/services/swagger_service.dart';
import 'package:swagger_mobile_client/widgets/welcome_page/error_text_widget.dart'
;
import 'package:swagger_mobile_client/widgets/welcome_page/welcome_widget.dart';

class WelcomePage extends StatefulWidget {
  @override
```

```
    _WelcomePageState createState() => _WelcomePageState();
}

class _WelcomePageState extends State<WelcomePage> {
  final SwaggerService _swaggerService = SwaggerService();
  final _urlController = TextEditingController();
  final _swaggerEndpointController = TextEditingController(
    text: StringConsts.swaggerEndpointHint,
  );

  bool _isLoading = false;
  bool _isErrorOccured = false;
  String errorMessage = "";

  onConfirmButtonClicked(BuildContext context) async {
    setState(()
    {
       _isErrorOccured = false;
      _isLoading = true;
    });

    final url = _urlController.text;
    final swaggerEndpoint = _swaggerEndpointController.text;

    try {
      final swaggerModel = await _swaggerService.getSwaggerModel(
        url,
        swaggerEndpoint,
      );

      Navigator.pushNamed(
        context,
        RouteConsts.controllersPage,
        arguments: swaggerModel,
      );
    } catch (e) {
      showErrorText(e);
    } finally {
      setState(() => _isLoading = false);
    }
  }

  void showErrorText(e) {
    _isErrorOccured = true;
    _isLoading = false;
    errorMessage = e.toString();
    print(e);
  }
```

```
  Widget errorText() {
    return _isErrorOccured
        ? ErrorText(errorMessage: "${StringConsts.errorText}\n$errorMessage")
        : SizedBox.shrink();
  }

  Widget loader() {
    return _isLoading
        ? Container(
            padding: EdgeInsets.only(top: 5),
            child: SpinKitFadingCircle(color: Colors.green, size: 50.0),
          )
        : SizedBox.shrink();
  }

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      builder: (context, snapshot) {
        return Scaffold(
          body: Container(
            padding: EdgeInsets.all(10),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: <Widget>[
                Welcome(
                  urlController: _urlController,
                  swaggerEndpointController: _swaggerEndpointController,
                  confirmButtonClickedHandler: () =>
                      onConfirmButtonClicked(context),
                ),
                errorText(),
                loader()
              ],
            ),
          ),
        );
      },
    );
  }
}
```

Файл error_text_widget.dart

```
import 'package:flutter/material.dart';
```

```
class ErrorText extends StatelessWidget {
  final String errorMessage;

  ErrorText({this.errorMessage});

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(
        errorMessage,
        style: TextStyle(color: Colors.red),
      ),
    );
  }
}
```

Файл welcome_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/custom_text_form_fi
eld_widget.dart';

class Welcome extends StatelessWidget {
  final TextEditingController urlController;
  final TextEditingController swaggerEndpointController;
  final Function confirmButtonClickedHandler;
  Welcome({
    this.urlController,
    this.confirmButtonClickedHandler,
    this.swaggerEndpointController,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(5),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text(
            StringConsts.enterUrl,
            style: const TextStyle(fontSize: 40),
          ),
          Container(
            child: Column(
              children: [
                CustomTextFormField(
```

```
                label: StringConsts.urlLabel,
                hint: StringConsts.urlHint,
                controller: urlController,
              ),
              CustomTextFormField(
                label: StringConsts.swaggerEndpointLabel,
                hint: StringConsts.swaggerEndpointHint,
                controller: swaggerEndpointController,
              ),
            ],
          ),
          width: 300,
          margin: const EdgeInsets.symmetric(vertical: 8),
        ),
        RaisedButton(
          color: Theme.of(context).buttonTheme.colorScheme.primary,
          textTheme: Theme.of(context).buttonTheme.textTheme,
          child: const Text(StringConsts.confirmBtn),
          onPressed: confirmButtonClickedHandler,
        ),
      ],
    ),
  );
 }
}
```

Файл authorize_button_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/security_schemes_model.d
art';
import 'package:swagger_mobile_client/widgets/controllers_page/authorize/authorize
_dialog_widget.dart';

class AuthorizeButton extends StatefulWidget {
  final SecuritySchemes securitySchemes;

  const AuthorizeButton(this.securitySchemes);

  @override
  _AuthorizeButtonState createState() => _AuthorizeButtonState();
}

class _AuthorizeButtonState extends State<AuthorizeButton> {
  bool isAuthorized = false;

  void _showAuthorizePopup(BuildContext ctx) {
```

```
      showDialog(
        context: ctx,
        builder: (_) => AuthorizeDialog(widget.securitySchemes, _updateAuthorization
),
        barrierDismissible: true,
      );
    }

    void _updateAuthorization(String value) {
      widget.securitySchemes.authorization.value = value;
      setState(() {
        isAuthorized = value.isNotEmpty;
      });
    }

    @override
    Widget build(BuildContext context) {
      return Align(
        alignment: Alignment.centerRight,
        child: Container(
          child: RaisedButton.icon(
            icon: Icon(isAuthorized ? Icons.lock_open : Icons.lock),
            onPressed: () => _showAuthorizePopup(context),
            shape: const RoundedRectangleBorder(
              borderRadius: const BorderRadius.all(
                const Radius.circular(15),
              ),
            ),
            label: Text(
                isAuthorized ? StringConsts.authorized : StringConsts.authorize),
            color: isAuthorized ? Colors.green : Colors.red,
          ),
        ),
      );
    }
}
```

Файл authorize_dialog_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/security_schemes_model.d
art';
import 'package:swagger_mobile_client/widgets/controllers_page/custom_text_form_fi
eld_widget.dart';

class AuthorizeDialog extends StatelessWidget {
  final TextEditingController controller = TextEditingController();
```

```
final SecuritySchemes securitySchemes;
final void Function(String) updateAuthorization;

AuthorizeDialog(this.securitySchemes, this.updateAuthorization) {
  controller.text = this.securitySchemes.authorization.value ?? "";
}

_closeDialog(BuildContext context) => Navigator.of(context).pop();

_cancel(BuildContext context) => _closeDialog(context);

_ok(BuildContext context) {
  this.updateAuthorization(controller.text);
  _closeDialog(context);
}

@override
Widget build(BuildContext context) {
  final key = this.securitySchemes.schemes.entries.first.key;
  final schema = this.securitySchemes.schemes.entries.first.value;

  return AlertDialog(
    title: Text(key),
    content: Container(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(schema.description),
          TextField(
            decoration: InputDecoration(
              hintText: StringConsts.authorization,
            ),
            controller: controller,
            keyboardType: TextInputType.multiline,
            maxLines: 3,
          ),
        ],
      ),
    ),
    actions: [
      FlatButton(
        child: Text(StringConsts.cancelBtn),
        onPressed: () => _cancel(context),
      ),
      FlatButton(
        child: Text(StringConsts.okBtn),
        onPressed: () => _ok(context),
      ),
```

```
        ],
      );
    }
  }
```

Файл array_parameter_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/array_parameter_model.dart';
import 'package:swagger_mobile_client/models/parameter_creation_model.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/services/parameter_creator.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/parameters_group_wi
dget.dart';
import 'package:swagger_mobile_client/widgets/non_scrollable_list_widget.dart';

class ArrayParameter extends StatefulWidget {
  final SwaggerParameterSchema parameter;
  final Function(List<dynamic>) onChanged;
  final ParameterCreator service = ParameterCreator();

  ArrayParameter({@required this.parameter, @required this.onChanged});

  @override
  _ArrayParameterState createState() => _ArrayParameterState();
}

class _ArrayParameterState extends State<ArrayParameter> {
  Map<String, ArrayParameterModel> values = Map();

  widgetChanged() {
    return widget.onChanged(values.entries.map((e) => e.value.value).toList());
  }

  void _removeItem(String id) {
    setState(() => values.removeWhere((key, value) => key == id));
    widgetChanged();
  }

  void _addItem() {
    final value = generateWidget(UniqueKey().toString());
    setState(() => values.addEntries([value]));
    widgetChanged();
  }

  void onChangedHandler(String id, dynamic value) {
```

```
    values[id].value = value;
    widgetChanged();
  }

  MapEntry<String, ArrayParameterModel> generateWidget(String id) {
    final w = Stack(
      overflow: Overflow.visible,
      children: <Widget>[
        widget.service.generateParameterUI(
          ParameterCreationModel(
            name: id,
            schema: widget.parameter.arraySchema,
            onChangedHandler: onChangedHandler,
          ),
        ),
        Positioned(
          right: -15,
          top: 0,
          child: SizedBox(
            height: 30,
            child: RawMaterialButton(
              padding: const EdgeInsets.all(0),
              onPressed: () => _removeItem(id),
              elevation: 2.0,
              fillColor: Colors.red, // TODO: Fix Color
              child: const Icon(Icons.remove, color: Colors.white),
              shape: const CircleBorder(),
            ),
          ),
        ),
      ],
    );

    return MapEntry(id, ArrayParameterModel(widget: w));
  }

  @override
  Widget build(BuildContext context) {
    return ParametersGroup(
      child: NonScrollableList(
        itemBuilder: (BuildContext ctx, int index) {
          return values.values.toList()[index].widget;
        },
        itemCount: values.length,
      ),
      name: widget.parameter.name,
      isAddButton: true,
      onAddBtnClicked: () => _addItem(),
```

```
    );
  }
}
```

Файл controller_list_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/swagger_json/swagger_model.dart';
import 'package:swagger_mobile_client/models/swagger_ui_model.dart';
import 'package:swagger_mobile_client/services/swagger_service.dart';
import 'package:swagger_mobile_client/widgets/non_scrollable_list_widget.dart';
import './controller_widget.dart';

class ControllerList extends StatelessWidget {
  final SwaggerService service = SwaggerService();
  final SwaggerModel model;
  List<ControllerUIModel> _controlUIModels;

  ControllerList(this.model) {
    _controlUIModels = service.getControllerUIModel(model);
  }

  @override
  Widget build(BuildContext context) {
    return NonScrollableList(
      itemBuilder: (BuildContext ctx, int index) => Controller(
        url: model.url,
        authorization: model.securitySchemes?.authorization,
        controllerUIModel: _controlUIModels[index],
      ),
      itemCount: _controlUIModels.length,
    );
  }
}
```

Файл controller_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/swagger_ui_model.dart';
import './method_list_widget.dart';

class Controller extends StatefulWidget {
  final String url;
  final ValueNotifier<String> authorization;
  final ControllerUIModel controllerUIModel;
```

```
    const Controller({
      this.url,
      this.authorization,
      this.controllerUIModel,
    });

    @override
    _ControllerState createState() => _ControllerState(controllerUIModel);
}

class _ControllerState extends State<Controller> {
    bool _isMethodsVisible = false;
    ControllerUIModel _controllerUIModel;

    _ControllerState(ControllerUIModel controllerUIModel) {
      _controllerUIModel = controllerUIModel;
    }

    void _toggleMethods() {
      setState(() => _isMethodsVisible = !_isMethodsVisible);
    }

    @override
    Widget build(BuildContext context) {
      return Card(
        elevation: 2,
        color: Theme.of(context).backgroundColor,
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            InkWell(
              child: ListTile(
                trailing: Icon(!_isMethodsVisible
                    ? Icons.keyboard_arrow_down
                    : Icons.keyboard_arrow_up),
                title: Text(_controllerUIModel.controllerName),
                onTap: () => _toggleMethods(),
              ),
            ),
            Visibility(
              child: MethodList(
                url: widget.url,
                authorization: widget.authorization,
                methodUIModels: _controllerUIModel.methods,
              ),
              visible: _isMethodsVisible,
            ),
          ],
```

```
      ),
    );
  }
}
```

Файл custom_checkbox_widget.dart

```
import 'package:flutter/material.dart';

class CustomCheckbox extends StatefulWidget {
  final String name;
  final void Function(bool) onChanged;

  const CustomCheckbox({this.name, @required this.onChanged});

  @override
  _CustomCheckboxState createState() => _CustomCheckboxState();
}

class _CustomCheckboxState extends State<CustomCheckbox> {
  bool value = false;

  checkboxChanged(bool v) {
    setState(() => value = v);
    widget.onChanged(value);
  }

  @override
  Widget build(BuildContext context) {
    return CheckboxListTile(
      onChanged: (value) => checkboxChanged(value),
      value: value,
      title: Text(widget.name),
    );
  }
}
```

Файл custom_datepicker_widget.dart

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';

class CustomDateTimePicker extends StatefulWidget {
  final DateTime dateTime;
  final String name;
  final void Function(String) validator;
  final void Function(DateTime) onChanged;
```

```
    const CustomDateTimePicker({
      this.dateTime,
      this.name,
      this.validator,
      @required this.onChanged,
    });

    @override
    _CustomDateTimePickerState createState() =>
        _CustomDateTimePickerState(dateTime);
}

class _CustomDateTimePickerState extends State<CustomDateTimePicker> {
    DateTime _selectedDateTime;

    _CustomDateTimePickerState(this._selectedDateTime);

    Future _selectDateTime(BuildContext context) async {
      // Remove focusing so that after choosing date
      // we won't be jumping to the latest focused input
      FocusScope.of(context).unfocus();

      if (_selectedDateTime == null) _selectedDateTime = DateTime.now();

      final date = await showDatePicker(
        context: context,
        initialDate: _selectedDateTime,
        firstDate: DateTime(0),
        lastDate: DateTime(2101),
      );

      final time = await showTimePicker(
        context: context,
        initialTime: TimeOfDay.fromDateTime(_selectedDateTime),
      );

      if (date != null && date != _selectedDateTime && time != null) {
        final dateTime = DateTime(
          date.year,
          date.month,
          date.day,
          time.hour,
          time.minute,
        );
        setState(() => _selectedDateTime = dateTime);

        widget.onChanged(dateTime);
      }
```

```
  }

  String get _dateTimeString {
    return _selectedDateTime != null
        ? DateFormat("dd-MM-yyyy HH:mm").format(_selectedDateTime.toLocal())
        : "";
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: ListTile(
        title: Text(widget.name),
        subtitle: Text(_dateTimeString),
        trailing: RaisedButton.icon(
          color: Colors.green,
          onPressed: () => _selectDateTime(context),
          icon: const Icon(Icons.date_range),
          label: const Text(StringConsts.datePickerBtn),
        ),
      ),
    );
  }
}
```

Файл custom_text_form_field_widget.dart

```
import 'package:flutter/material.dart';

class CustomTextFormField extends StatelessWidget {
  final TextInputType textInputType;
  final Function(String) validator;
  final String label;
  final String hint;
  final Function(String) onChanged;

  TextEditingController controller;

  CustomTextFormField({
    this.textInputType,
    this.validator,
    this.label,
    this.hint,
    controller,
    this.onChanged,
  }) {
    this.controller = controller ?? TextEditingController();
  }
```

```
  @override
  Widget build(BuildContext context) {
    return TextFormField(
      controller: controller,
      onChanged: (value) => onChanged(value),
      keyboardType: textInputType,
      maxLines: 1,
      validator: validator,
      decoration: InputDecoration(
        labelText: label,
        hintText: hint ?? label,
        labelStyle: const TextStyle(
          decorationStyle: TextDecorationStyle.solid,
        ),
      ),
    );
  }
}
```

Файл header_widget.dart

```
import 'package:flutter/material.dart';

class Header extends StatelessWidget {
  final String _text;
  const Header(this._text);

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Text(
        _text,
        style: const TextStyle(
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
}
```

Файл method_details_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_ui_model.dart';
```

```
import 'package:swagger_mobile_client/widgets/controllers_page/parameters_widget.d
art';
import 'package:swagger_mobile_client/widgets/controllers_page/request_body_widget
.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/responses_widget.da
rt';

class MethodDetails extends StatelessWidget {
  final MethodUIDetailModel detail;
  final Function onExecuteClicked;
  final Function onParametersChanged;
  final Function onRequestBodyChanged;
  final ValueNotifier<bool> responseIsLoading;

  final _formKey = GlobalKey<FormState>();

  MethodDetails({this.detail, this.onExecuteClicked, this.onParametersChanged, thi
s.onRequestBodyChanged, this.responseIsLoading});

  execute() {
    if (_formKey.currentState.validate()) {
      print("everything is okay");
      onExecuteClicked();
    } else {
      print("no something worng");
    }
  }
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(5),
      child: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            Parameters(detail.parameters, onParametersChanged),
            RequestBody(detail.requestBody, onRequestBodyChanged),
            Responses(detail.responses, responseIsLoading),
            RaisedButton(
              color: Theme.of(context).buttonTheme.colorScheme.primary,
              textTheme: Theme.of(context).buttonTheme.textTheme,
              child: const Text(StringConsts.executeBtn),
              onPressed: () => execute(),
            ),
          ],
        ),
      ),
```

```
    );
  }
}
```

Файл method_list_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/swagger_ui_model.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/method_widget.dart'
;
import 'package:swagger_mobile_client/widgets/non_scrollable_list_widget.dart';

class MethodList extends StatelessWidget {
  final String url;
  final ValueNotifier<String> authorization;
  final List<MethodUIModel> methodUIModels;

  const MethodList({this.url, this.authorization, this.methodUIModels});

  @override
  Widget build(BuildContext context) {
    return NonScrollableList(
      itemCount: methodUIModels.length,
      itemBuilder: (BuildContext ctx, int index) => Method(
        url: url,
        authorization: authorization,
        methodUIModel: methodUIModels[index],
      ),
    );
  }
}
```

Файл method_widget.dart

```
import 'package:flutter/material.dart';

import 'package:swagger_mobile_client/models/swagger_ui_model.dart';
import 'package:swagger_mobile_client/services/method_service.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/method_details_widg
et.dart';

class Method extends StatefulWidget {
  final String url;
  final ValueNotifier<String> authorization;
  final MethodUIModel methodUIModel;

  MethodService methodService;
```

```
  Method({this.url, this.authorization, this.methodUIModel}) {
    methodService = MethodService(url);
  }

  @override
  _MethodState createState() => _MethodState();
}

class _MethodState extends State<Method> {
  bool _isMethodDetailsVisible = false;

  Map<String, dynamic> parameters;
  Map<String, dynamic> requestBody;

  // For showing loader while daata is being loaded
  ValueNotifier<bool> responseIsLoading = ValueNotifier(false);

  void executeClicked() async {
    responseIsLoading.value = true;

    final response = await widget.methodService.sendRequest(
      model: widget.methodUIModel,
      parameters: parameters,
      requestBody: requestBody,
      authorizationValue: widget.authorization?.value
    );

    responseIsLoading.value = false;
    widget.methodUIModel.detail.responses.value = response;
  }

  void parametersChanged(Map<String, dynamic> data) {
    parameters = data;
  }

  void requestBodyChanged(Map<String, dynamic> data) {
    requestBody = data;
    print(requestBody);
  }

  void _showMethodDetails() {
    setState(() => _isMethodDetailsVisible = !_isMethodDetailsVisible);
  }

  @override
  Widget build(BuildContext context) {
    return Card(
```

```
        elevation: 2,
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            InkWell(
              child: ListTile(
                contentPadding: const EdgeInsets.symmetric(horizontal: 5),
                dense: true,
                leading: Container(
                  padding: const EdgeInsets.all(10),
                  decoration: BoxDecoration(
                    borderRadius: const BorderRadius.all(
                      const Radius.circular(5),
                    ),
                    color: widget.methodUIModel.typeColor,
                  ),
                  child: Text(
                    widget.methodUIModel.type.toUpperCase(),
                    style: const TextStyle(color: Colors.white),
                  ),
                ),
                trailing: Icon(!_isMethodDetailsVisible
                    ? Icons.keyboard_arrow_down
                    : Icons.keyboard_arrow_up),
                title: Text(widget.methodUIModel.endPoint),
                onTap: () => _showMethodDetails(),
              ),
            ),
            Visibility(
              child: MethodDetails(
                detail: widget.methodUIModel.detail,
                onExecuteClicked: executeClicked,
                onParametersChanged: parametersChanged,
                onRequestBodyChanged: requestBodyChanged,
                responseIsLoading: responseIsLoading,
              ),
              visible: _isMethodDetailsVisible,
            ),
          ],
        ),
      );
    }
}
```

Файл object_parameter_widget.dart

```
import 'package:flutter/material.dart';
```

```
import 'package:swagger_mobile_client/models/parameter_creation_model.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/services/parameter_creator.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/parameters_group_wi
dget.dart';
import 'package:swagger_mobile_client/widgets/non_scrollable_list_widget.dart';

class ObjectParameter extends StatelessWidget {
  final SwaggerParameterSchema parameter;
  final Function(String, dynamic) onChanged;
  final ParameterCreator service = ParameterCreator();
  Map<String, dynamic> values = Map();

  ObjectParameter({this.parameter, this.onChanged}) {
    values = service.initValuesMap(parameter?.childSchemas);
  }

  onChangedHandler(String name, dynamic value) {
    values[name] = value;
    onChanged(name, value);
  }

  Widget build(BuildContext context) {
    return ParametersGroup(
      child: NonScrollableList(
        itemBuilder: (BuildContext ctx, int index) {
          return service.generateParameterUI(
            ParameterCreationModel(
              schema: parameter.childSchemas[index],
              onChangedHandler: onChangedHandler,
            ),
          );
        },
        itemCount: parameter.childSchemas != null ? parameter.childSchemas.length
: 0,
      ),
      name: parameter.name,
      isAddButton: false,
    );
  }
}
```

Файл parameters_group_widget.dart

```
import 'package:flutter/material.dart';

class ParametersGroup extends StatelessWidget {
```

```
final String name;
final Widget child;
final Function onAddBtnClicked;

bool isAddButton = false;

Widget _createAddButton() {
  print(isAddButton);
  if (isAddButton) {
    return Positioned(
      right: -35,
      bottom: -15,
      child: RawMaterialButton(
        onPressed: () => onAddBtnClicked(),
        elevation: 2.0,
        fillColor: Colors.green, // TODO: Fix Color
        child: const Icon(Icons.add, color: Colors.white),
        shape: const CircleBorder(),
      ),
    );
  }
  return SizedBox.shrink();
}

ParametersGroup({
  this.child,
  this.name,
  this.isAddButton,
  this.onAddBtnClicked,
});

@override
Widget build(BuildContext context) {
  return Container(
    child: Stack(
      overflow: Overflow.visible,
      children: <Widget>[
        Container(
          padding: const EdgeInsets.symmetric(horizontal: 10),
          constraints: const BoxConstraints(minHeight: 50),
          margin: const EdgeInsets.symmetric(vertical: 10),
          decoration: BoxDecoration(
            border: Border.all(
              style: BorderStyle.solid,
              width: 1,
              color: Colors.grey,
            ),
            borderRadius: BorderRadius.circular(10),
```

```
            ),
            child: child,
          ),
          Positioned(
            left: 15,
            top: 0,
            child: Container(
              padding: const EdgeInsets.only(left: 10, right: 10),
              color: Colors.white,
              child: name == ''
                  ? SizedBox.shrink()
                  : Text(
                      name,
                      style: TextStyle(color: Colors.black, fontSize: 12),
                    ),
            ),
          ),
          _createAddButton(),
        ],
      ),
    );
  }
}
```

Файл parameters_template_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/parameter_creation_model.dart';

import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/services/parameter_creator.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/header_widget.dart'
;
import 'package:swagger_mobile_client/widgets/non_scrollable_list_widget.dart';

class ParametersTemplate extends StatelessWidget {
  final String header;
  final List<SwaggerParameterSchema> parameters;
  final void Function(Map<String, dynamic>) dataChanged;

  ParameterCreator service;
  Map<String, dynamic> values;

  ParametersTemplate({this.header, this.parameters, this.dataChanged}) {
    service = ParameterCreator();
    initValues();
  }
```

```
  void initValues() {
    values = service.initValuesMap(parameters);
    this.dataChanged(values);
  }

  void onChangedHandler(String name, dynamic value) {
    print("onchanged");
    print(name);
    values[name] = value;
    print(values);
    this.dataChanged(values);
  }

  @override
  Widget build(BuildContext context) {
    if (parameters == null || parameters.length == 0) return SizedBox.shrink();

    return Card(
      elevation: 2,
      child: Container(
        padding: const EdgeInsets.all(10),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            Header(header),
            NonScrollableList(
              itemBuilder: (BuildContext ctx, int index) {
                return service.generateParameterUI(ParameterCreationModel(
                  schema: parameters[index],
                  onChangedHandler: onChangedHandler,
                ));
              },
              itemCount: parameters.length,
            ),
          ],
        ),
      ),
    );
  }
}
```

Файл parameters_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
```

```
import 'package:swagger_mobile_client/services/parameter_creator.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/parameters_template
_widget.dart';

class Parameters extends StatelessWidget {
  final SwaggerParameters parameters;
  final ParameterCreator service = ParameterCreator();
  final void Function(Map<String, dynamic>) parametersChanged;
  Parameters(this.parameters, this.parametersChanged);

  @override
  Widget build(BuildContext context) {
    return ParametersTemplate(
      parameters: parameters?.content,
      header: StringConsts.parameters,
      dataChanged: parametersChanged
    );
  }
}
```

Файл request_body_widget.dart

```
import 'package:flutter/material.dart';
import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/services/parameter_creator.dart';
import 'package:swagger_mobile_client/widgets/controllers_page/parameters_template
_widget.dart';

class RequestBody extends StatelessWidget {
  final SwaggerRequestBody requestBody;
  final ParameterCreator service = ParameterCreator();
  final void Function(Map<String, dynamic>) requestBodyChanged;

  RequestBody(this.requestBody, this.requestBodyChanged);

  @override
  Widget build(BuildContext context) {
    return ParametersTemplate(
      parameters: requestBody == null ? null : [requestBody?.content],
      header: StringConsts.requestBody,
      dataChanged: requestBodyChanged,
    );
  }
}
```

Файл responses_widget.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_spinkit/flutter_spinkit.dart';

import 'package:swagger_mobile_client/models/consts/string_consts.dart';
import 'package:swagger_mobile_client/models/swagger_json/swagger_method_model.dar
t';
import 'package:swagger_mobile_client/widgets/controllers_page/header_widget.dart'
;

class Responses extends StatefulWidget {
  final ValueNotifier<Map<String, SwaggerResponse>> responses;
  final ValueNotifier<bool> responseLoading;

  Responses(this.responses, this.responseLoading);

  @override
  _ResponsesState createState() => _ResponsesState();
}

class _ResponsesState extends State<Responses> {
  Widget showLoading() {
    return Center(
      // TODO: Fix color
      child: Container(
        padding: EdgeInsets.symmetric(vertical: 10),
        child: SpinKitThreeBounce(
          color: Colors.green,
          size: 50,
        ),
      ),
    );
  }

  Widget showResponse() {
    return ValueListenableBuilder(
      valueListenable: widget.responses,
      builder: (context, responses, _) {
        return Container(
          padding: const EdgeInsets.all(10),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              const Header(StringConsts.responses),
              ListView.builder(
                itemCount: responses.length,
                scrollDirection: Axis.vertical,
                shrinkWrap: true,
                physics: const NeverScrollableScrollPhysics(),
```

```
                itemBuilder: (BuildContext context, int index) {
                  final response = responses.values.toList()[index];
                  return Column(
                    children: <Widget>[
                      Container(
                        margin: const EdgeInsets.only(top: 10, bottom: 0),
                        child: Row(
                          crossAxisAlignment: CrossAxisAlignment.start,
                          children: <Widget>[
                            Column(
                              children: <Widget>[
                                Text(responses.keys.toList()[index]),
                                Text(response.description),
                              ],
                            ),
                            Expanded(
                              child: Container(
                                margin: EdgeInsets.only(left: 5),
                                child: SelectableText(response.content),
                              ),
                            ),
                          ],
                        ),
                      ),
                    ],
                  );
                },
              ),
            ],
          ),
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    print(widget.responseLoading);
    return Card(
      elevation: 2,
      child: ValueListenableBuilder(
        valueListenable: widget.responseLoading,
        builder: (context, value, _) {
          if (value) {
            return showLoading();
          } else {
            return showResponse();
          }
```

```
          },
        ),
      );
    }
  }
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Програма та методика тестування

КПІ.ІП-6314.045440.05.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Д.О. Нечай

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ М.В. Карпа

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є мобільний застосунок для операційних систем на базі Android версії не нижче 6.0, що призначений для автоматизації роботи з HTTP API запитами на основі метаданих.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення у відповідно до функціональних вимог;

– знаходження проблем та недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування - перевіряється уся документація, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування - застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на безлічі тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми в роботі програми;

– тестування «білої скриньки» - об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності в користуванні. Для того, щоб перевірити працездатність та відмовостійкість додатку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на мобільних пристроях з різною роздільною здатністю екрану;

– тестування на мобільних пристроях з різною версією операційної системи;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування зміни орієнтації екрану;

– тестування працездатності програми у випадку відсутності з’єднання до мережі;

– тестування інтерфейсу користувача;

– тестування зручності використання.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Керівництво користувача

КПІ.ІП-6314.045440.06.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Д. О. Нечай

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ М. В. Карпа

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

«Swagger Mobile Client» - це мобільний застосунок для операційних систем на базі «Android». Даний додаток призначений для полегшення та автоматизації роботи з HTTP API запитами на мобільних платформах.

Користувач має можливість отримати повну інформацію про контролери, методи та їх параметри необхідного веб-сервісу. Він може ввести дані в параметри певного методу, провести їх валідацію та надіслати запит з цими даними на віддалений сервер. Користувач також має можливість вводити токен авторизації у випадку, якщо необхідна автентифікація для виконання запиту.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність мобільного пристрою з операційною системою на базі Android з версією 6.0 або вище;

– наявність доступу до Інтернету;

– для встановлення додатку на мобільному пристрої повинно бути не менше 70 МБ вільної пам’яті.

2.2 Завантаження застосунку

На даний момент застосунок можна встановити власноруч, використовуючи відповідний apk-файл. Для цього спершу необхідно завантажити його на мобільний пристрій, а потім, використовуючи який-небудь інсталятор, виконати встановлення даного додатку.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з’явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись початкова сторінка застосунку.

3 РОБОТА ІЗ ЗАСТОСУНКОМ

При запуску програмного застосунку користувачу буде відображено два поля для вводу url-адреси сервера та OpenAPI кінцеву точку, яку використовує даний сервер (рис. 3.1).

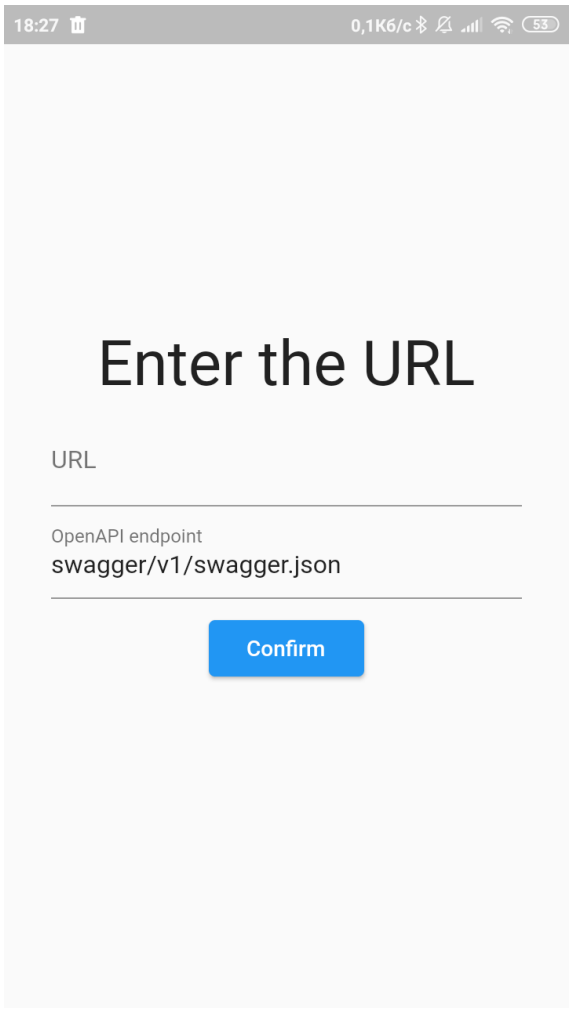


Рисунок 3.1 – Початкова сторінка додатку

Користувач має змогу ввести адресу сервера у відповідне поле та натиснути на кнопку «Confirm» для отримання метаданих. Після натискання на кнопку, буде зображено індикатор, що сповіщатиме про виконання запиту (рис. 3.2).

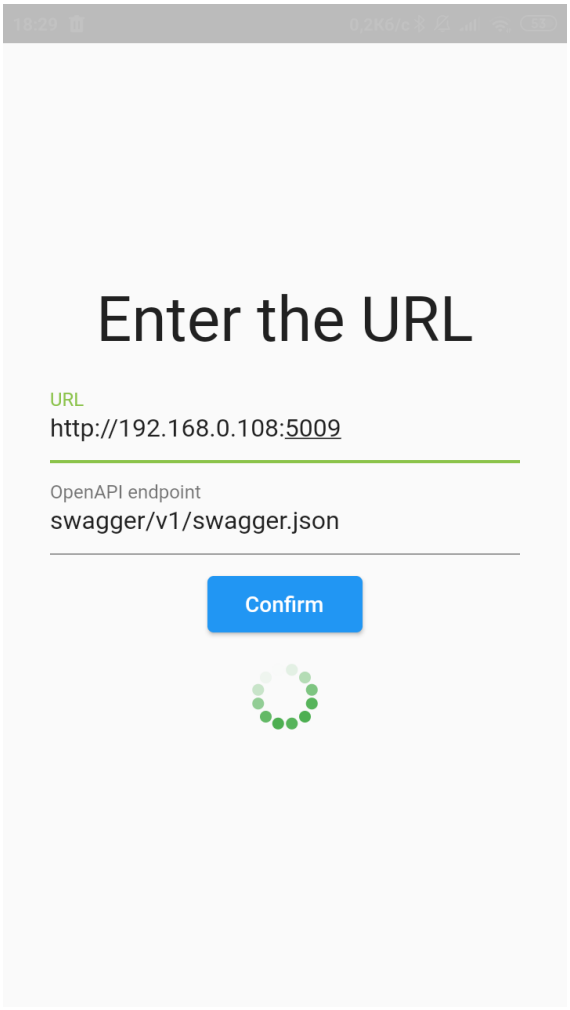


Рисунок 3.2 – Індикатор повідомляє про виконання запиту

У разі, якщо виникне помилка, відповідне повідомлення буде зображено в полі, де раніше працював індикатор (рис. 3.3).

Рисунок 3.3 – Повідомлення про помилку

У разі ж, якщо помилки не було, метадані будуть оброблені. На основі них буде побудований графічний інтерфейс. Відкриється наступна сторінка, на якій відображено список усіх контролерів (рис 3.4).

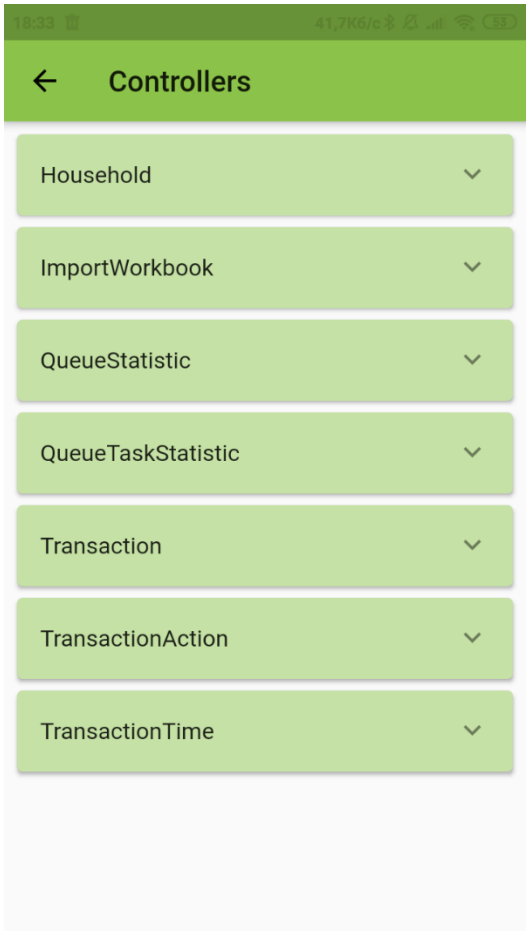


Рисунок 3.4 – Список контролерів

Користувач має змогу натиснути на потрібний контролер для отримання інформації про його методи. Після натискання, контролер «розгорнеться» і буде відображено список його методів (рис 3.5).

Рисунок 3.5 – Список методів вибраного контролера

Користувач має змогу «розгорнути» декілька контролерів одночасно. Якщо кількість елементів настільки велика, що вони не вміщуються в розміри екрану, користувач зможе прокрутити список до потрібного елементу (рис. 3.6).

Рисунок 3.6 – Декілька відкритих контролерів

Після натискання на потрібний метод, буде відображено список його вхідних параметрів (рис 3.7).

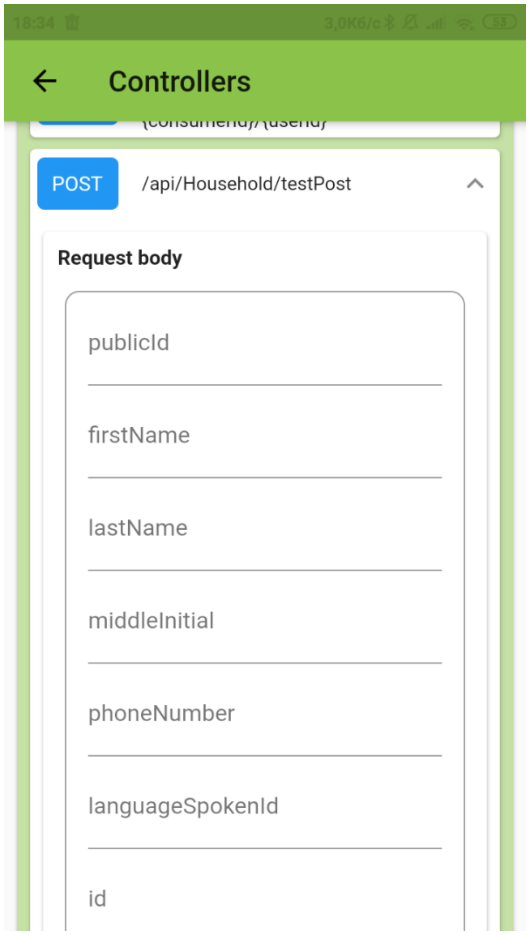


Рисунок 3.7 – Список параметрів певного методу

Натиснувши на кнопку «Execute», буде здійснено валідацію вхідних параметрів. У випадку, якщо параметри є неправильними, відповідне повідомлення буде відображено користувачу під даним параметром (рис 3.8).

Рисунок 3.8 – Валідація даних

У залежності від типу даних відповідний графічний об’єкт буде створений. Наприклад: якщо тип об’єкту – дата, то буде створена кнопка, при натисканні на яку, користувачу буде відображено вбудований в ОС календар, на якому значно зручніше вибирати дату та час, аніж вручну вписувати в текстовий рядок (рис 3.9).

Рисунок 3.9 – Календар для обрання дати та часу

У разі, якщо усі дані вірні, буде надіслано відповідний запит на віддалений веб-сервер. Відповідь, яка від нього прийде буде відображена у відповідному полі «Responses» (рис 3.10).

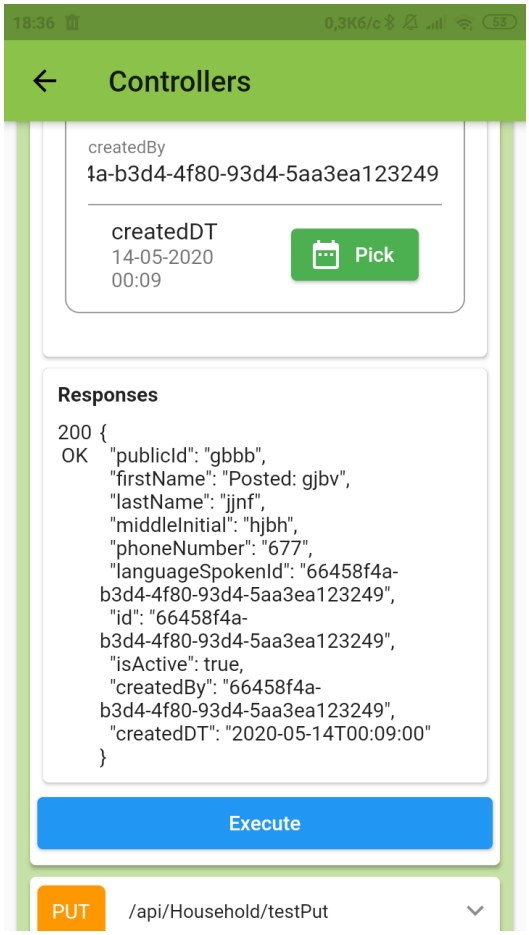


Рисунок 3.10 – Відповідь від сервера

У разі, якщо для успішного виконання запитів необхідна автентифікація користувача, буде побудована необхідна кнопка «Authorize» (рис 3.11).

Рисунок 3.11 – Кнопка «Authorize»

Спробувавши надіслати запит, не авторизувавшись, користувач отримає відповідь з кодом помилки 401 «Unauthorized» (рис 3.12).

Рисунок 3.12 – Відповідь «Unathorized»

Якщо натиснути на кнопку «Authorize», користувачу буде відображено модальне вікно, де він має змогу отримати інформацію про структуру необхідного токена та ввести цей токен (рис 3.13).

Рисунок 3.13 – Діалогове вікно для введення токена авторизації

Після того, як токен буде введений, кнопка змінить свій стиль та назву, що повідомлятиме користувача про те, що токен не є пустим. І тепер, у разі, якщо дані введено вірно, користувач має змогу виконувати запити на методи, що потребують автентифікації.

Рисунок 3.14 – Виконання методу, що потребує авторизації

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

МОБІЛЬНЕ ЗАСТОСУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З HTTP API НА ОСНОВІ МЕТАДАНИХ

Графічний матеріал

КПІ.ІП-6314.045440.07.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Д.О. Нечай

Нормоконтроль:

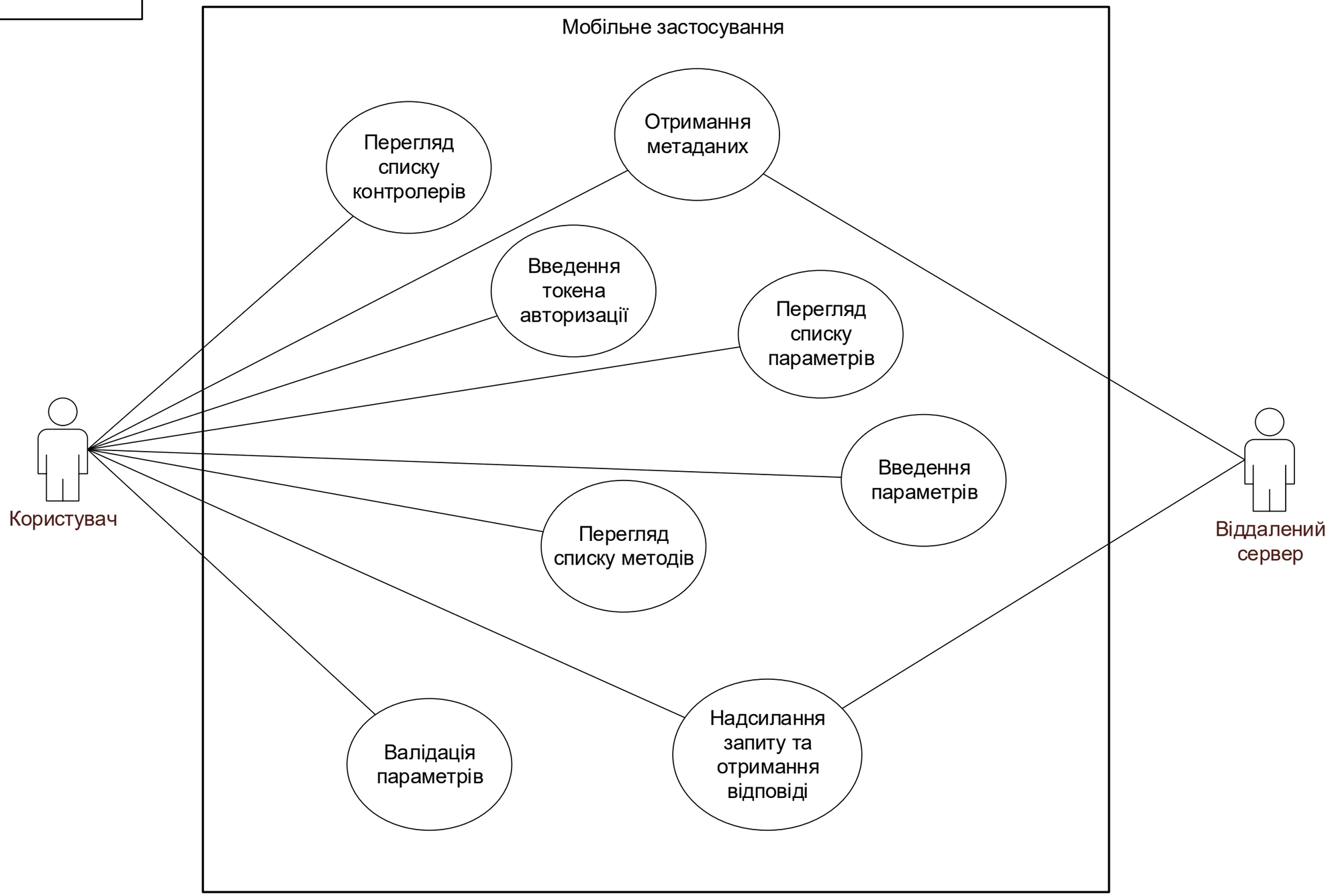
_____ К.І. Ліщук

Виконавець:

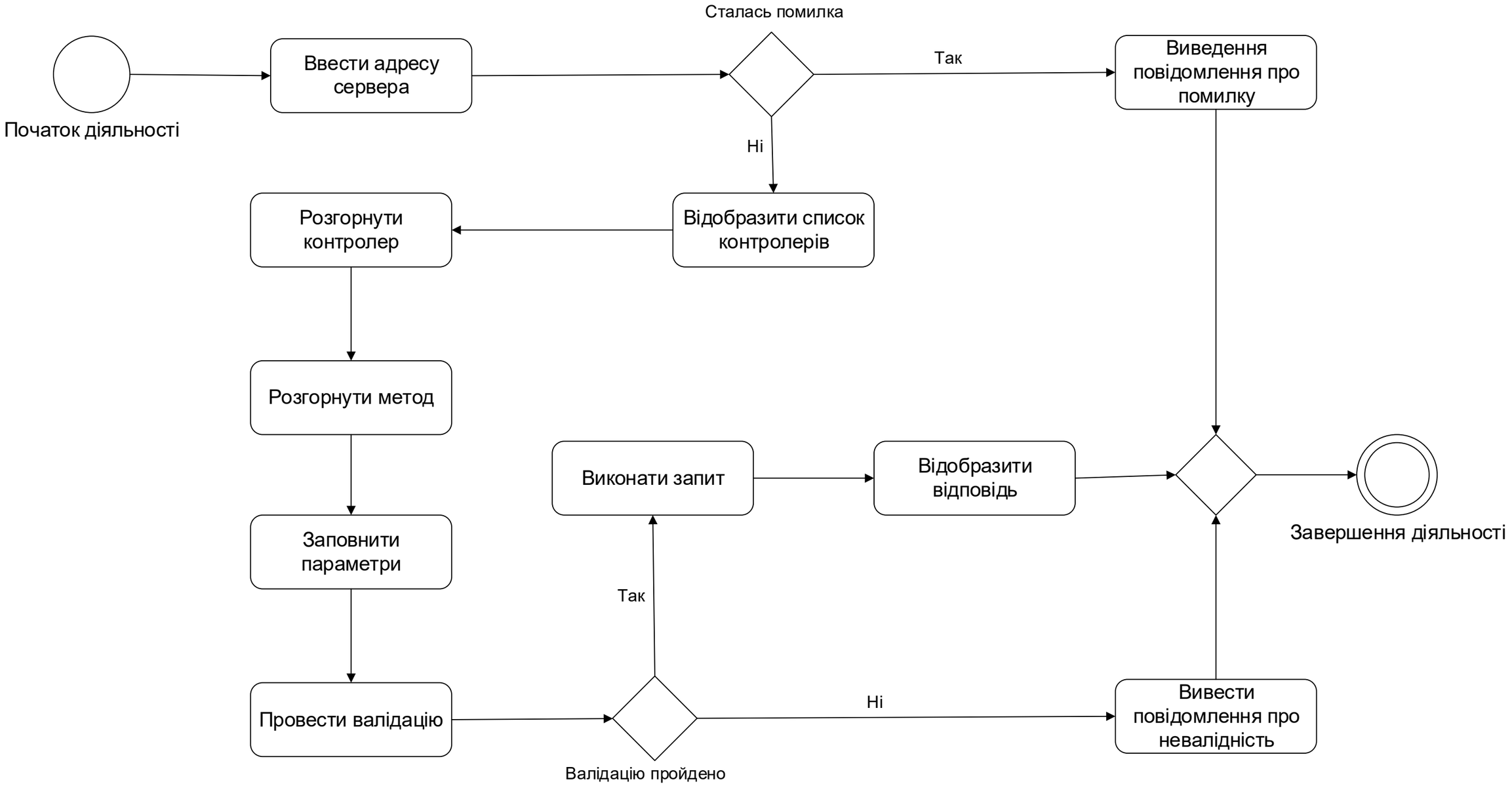
_____М.В. Карпа

Київ – 2020 року

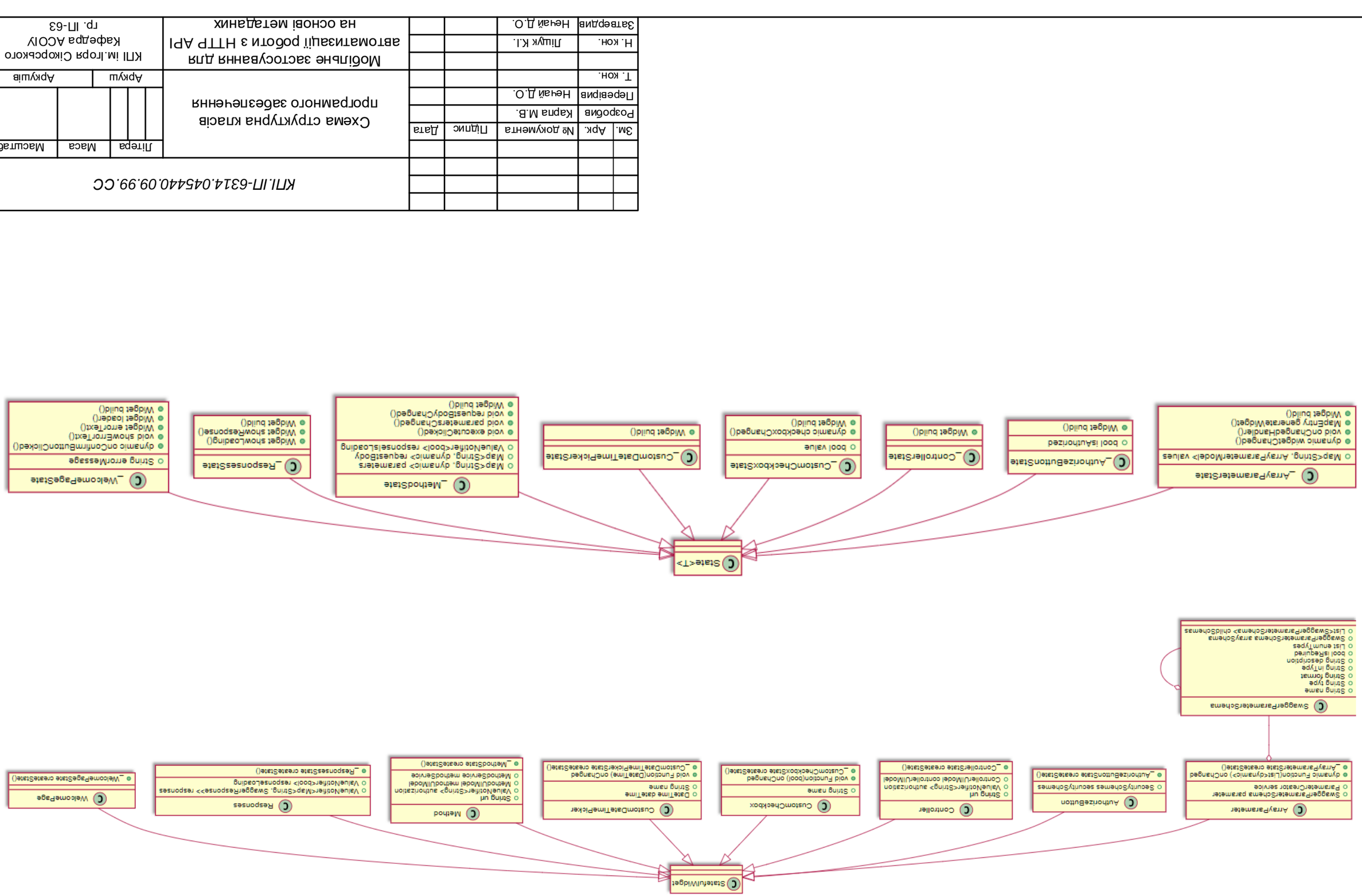
КПІ.ІП-6314.045440.07.99.СС



					КПІ.ІП-6314.045440.07.99.СС		
					Схема структурна варіантів використання	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Карпа М.В.				Аркуш	Аркушів
Перевірив		Нечай Д.О.			Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63	
Т. кон.							
Н. кон.		Ліщук К.І.					
Затвердив		Нечай Д.О.					



					КПІ.ІП-6314.045440.08.99.СС		
					Схема структурна діяльності	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Карпа М.В.				Аркуш	Аркушів
Перевірив		Нечай Д.О.					
Т. кон.							
					Мобільне застосування для автоматизації роботи з HTTP API на основі метаданих	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63	
Н. кон.		Ліщук К.І.					
Затвердив		Нечай Д.О.					


StatefulWidget
State<T>
ArrayParameterState
Map<String, ArrayParameterModel> values
dynamic widgetChanged()
void onChangedHandler()
MapEntry generateWidget()
Widget build()
AuthorizeButtonState
bool isAuthorized
Widget build()
ControllerState
Widget build()
CustomCheckboxState
bool value
dynamic checkboxChanged()
Widget build()
CustomDateTimePickerState
Widget build()
MethodState
Map<String, dynamic> parameters
Map<String, dynamic> requestBody
ValueNotifier<bool> responseIsLoading
void executeClicked()
void parametersChanged()
void requestBodyChanged()
Widget build()
ResponsesState
Widget showLoading()
Widget showResponse()
Widget build()
WelcomePageState
String errorMessage
dynamic onConfirmButtonClicked()
Widget errorText()
void showErrorText()
Widget loader()
Widget build()
ArrayParameter
SwaggerParameterSchema parameter
ParameterCreator service
dynamic Function(List<dynamic>) onChanged
ArrayParameterState createState()
AuthorizeButton
SecuritySchemes securitySchemes
AuthorizeButtonState createState()
Controller
String url
ValueNotifier<String> authorization
ControllerUIModel controllerUIModel
ControllerState createState()
CustomCheckbox
String name
void Function(bool) onChanged
CustomCheckboxState createState()
CustomDateTimePicker
DateTime dateTime
String name
void Function(DateTime) onChanged
CustomDateTimePickerState createState()
Method
String url
ValueNotifier<String> authorization
MethodUIModel methodUIModel
MethodService methodService
MethodState createState()
Responses
ValueNotifier<Map<String, SwaggerResponse>> responses
ValueNotifier<bool> responseLoading
ResponsesState createState()
WelcomePage
WelcomePageState createState()
SwaggerParameterSchema
String name
String type
String format
String inType
String description
bool isRequired
List enumTypes
SwaggerParameterSchema arraySchema
List<SwaggerParameterSchema> childSchemas